MCFPTree: An FP-tree-based algorithm for multi-constraint patterns discovery

Wen-Yang Lin*

Department of Computer Science and Information Engineering,
National University of Kaohsiung,
Kaohsiung 811, Taiwan, ROC
E-mail: wylin@nuk.edu.tw
*Corresponding author

Ko-Wei Huang

Institute of Computer and Communication Engineering,
National Cheng Kung University,
Tainan 701, Taiwan, ROC
E-mail: elone530@gmail.com

Chin-Ang Wu

Institute of Information Engineering,
I-Shou University,
Kaohsiung 840, Taiwan, ROC
E-mail: cwu@csu.edu.tw

Abstract: In this paper, the problem of constraint-based pattern discovery is investigated. By allowing more user-specified constraints other than traditional rule measurements, e.g., minimum support and minimum confidence, research work on this topic endeavoured to reflect real interest of analysts and relieve them from the overabundance of rules. Surprisingly, very little research has been conducted to deal with multiple types of constraints. In our previous work, we have studied this problem, specifically focusing on three different types of constraints, and an efficient Apriori-like algorithm, called MCFP, is proposed. In this paper, we propose a new algorithm called MCFPTree, which is based on a tree structure for keeping frequent patterns without suffering from the problem of candidate itemsets generation. Experimental results show that our MCFPTree algorithm is significantly faster than MCFP and an intuitive method FP-Growth+, i.e., post-processing the frequent patterns generated by FP-Growth, against user-specified constraints.

Keywords: multi-constraint pattern mining; item constraint; aggregation constraint; FP-tree.

Reference to this paper should be made as follows: Lin, W-Y., Huang, K-W. and Wu, C-A. (xxxx) 'MCFPTree: An FP-Tree-based algorithm for multi-constraint patterns discovery', *Int. J. Business Intelligence and Data Mining*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Wen-Yang Lin is a Professor in the Department of Computer Science and Information Engineering, National University of Kaohsiung. He is also the Director of Library and Information Center of the University. His research interests include data warehousing, data mining and evolutionary computation. He is also interested in the area of sparse matrix technology and large-scale supercomputing. He is regular reviewer for major journals and conferences, and has co-edited several special issues of renowned international journals, (co-)authored more than 120 refereed publications, served as co-chair, a member of programme committees and organised special sessions for many international conferences.

Ko-Wei Huang received an MS in Computer Science and Information Engineering from National University of Kaohsiung in 2008. Currently, he is a PhD student of Institute of Computer and Communication Engineering, National Cheng Kung University. His research interests include data mining, knowledge engineering and web intelligence.

Chin-Ang Wu is a PhD student in the Department of Information Engineering at the I-Shou University, Taiwan. She is also a Lecturer in Cheng Shiu University, Taiwan. She received her MS in Computer Science from George Washington University, Washington DC, USA, in 1988. Her research interests include data mining, data warehousing and database systems.

1 Introduction

Mining association rules from a database or data warehouse has been one of the main data-mining techniques supported by business intelligence systems to help a decision-maker acquire a better understanding of its commercial context. In the early stage of the development of association mining algorithms, most researches were devoted to designing efficient algorithms for generating frequent itemsets, ignoring the fact that lots of frequent patterns generated are not user-interested. This leads to the development of constraint-based association mining (Bayardo Jr. et al., 2000; Bonchi et al., 2003; Bonchi and Lucchese, 2007; Ceglar and Roddick, 2007; Cho et al., 2005; De Raedt and Zimmermann, 2007; Grahne et al., 2000; Lee et al., 2006; Lin et al., 2008; Lu et al., 2005; Ng et al., 1998; Pei and Han, 2000; Pei et al., 2004; Song and Qin, 2005; Srikant et al., 1997). By allowing user-specified constraints other than minimum support and minimum confidence, the discovered patterns can reflect real interest of the analysts and, in this way, can relieve them from the overabundance of rules.

So far, most work on constraint-based association mining has been single-constraint-oriented, i.e., only one type of constraint is considered. Surprisingly, little research has been conducted to deal with multiple types of constraints. This motivates us to the study of multi-constraint-based frequent pattern mining.

In our previous work (Lin et al., 2008), we have investigated this problem considering three different types of constraints, including item constraint, aggregation constraint and cardinality constraint. We also have proposed an efficient Apriori-like algorithm, called MCFP, for accomplishing this task. The proposed MCPF algorithm, however, has two significant weak points: First, its performance is heavily affected by the size of item constraint and the number of aggregation constraints, owing to the paradigm of candidate

generation and pruning technique it adopts. Second, it cannot deal with item constraints containing negative items.

In this regard, we proposed a new algorithm called MCFPTree. The main advantage of MCFPTree over MCFP is that MCFPTree is an FP-Growth-like algorithm. By adopting the FP-tree structure, it heavily eliminates the load of candidate generation. Besides, by incorporating a new approach for exploiting item constraint, MCFPTree can handle item constraints that contain negative items.

An experiment on both synthetic data set and real data set show that our MCFPTree algorithm is significantly faster than MCFP and an intuitive method FP-Growth+, i.e., post-processing the frequent patterns generated by FP-Growth (Han et al., 2000), against user-specified constraints.

The remainder of this paper is organised as follows. The background knowledge and related work is described in Section 2. Section 3 defines some terminologies, formalises the problem of constraint-based association rules mining, and describes the proposed algorithm, MCFTPree. Evaluations of our MCFPTree algorithm against MCFP and FP-Growth+ on both synthetic data set and real data set are described in Section 4. Finally, conclusions and future work are stated in Section 5.

2 Background and related work

Traditional techniques for mining association rules may generate thousands of rules or frequent patterns. Unfortunately, most of which are uninteresting to the users; only a relatively small subset of the complete frequent patterns and association rules is of interest to users. In light of these, constraint-based techniques have been developed for mining frequent patterns and association rules.

The main purpose of constraint-based mining is to let users specify what kinds of knowledge or patterns that really are of interest to guide the mining methods to search for useful patterns, rather than spending much time on discovering patterns that the users have no interest. According to Han and Kamber (2004), there are five different categories of constraint.

- *Knowledge-type constraints*: This refers to the type of knowledge to be mined, such as the association rules and classification.
- *Data constraints*: This refers to the set of the task-relevant data, such as about the bookstore sales mining.
- *Dimensional/level constraints*: This refers to the desired dimensions of data, or levels of the concept hierarchies.
- *Interestingness constraints*: This refers to the measures of rule interestingness, such as minimum support and minimum confidence.
- *Rule constraints*: This refers to the form of rules to be mined, such as aggregation constraints, meta-rule and item constraint.

Our study in this paper is focused on the rule constraints. Specifically, three different types of rule constraints are considered, including item constraint, aggregation constraint and cardinality constraint.

2.1 Item constraint

The concept of item constraint was first proposed by Srikant et al. (1997), which is expressed in the form of a Boolean expression, indicating the presence or absence of items that are interesting to the users. Specifically, an item constraint expressed as a Boolean expression is in disjunctive normal form $D_1 \vee D_2 \vee \ldots \vee D_m$, where each disjunct D_i is of the form of the $a_1 \wedge a_2 \wedge \ldots \wedge a_n$, and each element a_i is either l_i or $\sim l_i$ for some $l_i \in \boldsymbol{I}$. For example, the Boolean expression $(a \wedge b) \vee (c \wedge \sim d)$ denotes that the user is interested to see any patterns containing both items *a* and *b* or any patterns containing items *c* but not *d*. In the context of business intelligence, the item constraint specification, for example, would help a sales manager focus only on association exploration involving new products to understand the effectiveness of sales strategy for promoting these products.

Srikant et al. (1997) also proposed three algorithms to accomplish the task of mining frequent itemsets that satisfy a given item constraint, called Multiple Joins, Reorder and Direct. All of these methods are developed following the classical Apriori framework, i.e., a level-wise, bottom-up generation and inspection of candidate itemsets, but each differing in the procedure for generating next $(k + 1)$-level candidates from previous *k*-level frequent patterns. However, no implementation and empirical evaluation of the proposed algorithms were conducted.

Lu et al. (2005) introduced an ECLAT-based algorithm, Eclat II, which is featured by pushing the Boolean expression item constraint into the ECLAT framework (Zaki, 2000). Their work also provided no algorithmic implementation and empirical evaluation.

2.2 Aggregation constraint

An aggregation constraint is a constraint defined on an aggregation function of itemsets, such as *avg*, *sum* and *median*. An aggregation constraint is of the form $avg(S)\theta v$, $sum(S)\theta\ v$, or $median(S)\theta v$, where S is an itemset, v a real value, and θ *is* a comparison operator, i.e., $\theta \in \{\geq, >, \leq, <\}$. For example, $avg(S) \geq 30$ or $sum(S) \leq 20$. In real applications, this type of constraint specification provides the facility to discover the patterns as a whole satisfying some characteristics. For example, suppose that a retailing manager wants to find associations containing expensive items from customer transactions. The manager may specify some constraints that involve an item with price more than \$300, and the total amount of each pattern should be at least \$1000.

According to the study conducted by Ng et al. (1998), Pei and Han (2000), Pei et al. (2004) and Grahne et al. (2000), the aggregation constraints can be classified as *anti-monotone*, *monotone* and *succinct*. A constraint C_{am} is anti-monotone if and only if whenever an itemset S violates C_{am}, so does any superset of S. A constraint C_m is monotone if and only if whenever an itemset S satisfies C_m, so does any superset of S. A constraint C_s is succinct if given A_i, the set of items satisfying C_s, then any set S satisfying C_s is based on A_i, i.e., S contains a subset belonging to A_i. For example, $avg(S) \leq 20$ is an anti-monotone aggregation constraint, $sum(S) \geq 20$ is a monotone aggregation constraint, and $min(S) \geq 30$ is a succinct aggregation constraint, supposed that some of items have non-negative values. Besides, an aggregation constraint can be *convertible* or *inconvertible*. An aggregation constraint C_{ag} is convertible provided there is an order R (ascending or descending) on items such that constraint is convertible anti-monotone or convertible monotone; otherwise, we say

that aggregation constraint is inconvertible. Ng et al. (1998) first proposed an Apriori-based mining algorithm CAP for handling anti-monotone and succinct constraints. Pei and Han (2000) then proposed a new constraint-based frequent pattern-mining algorithm called CFG. The method of mining association rules in large, dense databases by user-defined constraints was studied by Bayardo Jr. et al. (2000). Constraint-based mining of correlations, by exploration of anti-monotone, succinct and monotone constraints, was developed in Grahne et al. (2000). Bonchi et al. (2003) proposed an Apriori-like algorithm that exploits anti-monotone and monotone constraints to reduce the problem dimensions. A more general framework then was developed by De Raedt and Zimmermann (2007), which accommodates other constraint functions, such as redundancy, representativeness and top-*k* mining.

The technique of pushing convertible aggregation constraint into association mining algorithm was first studied by Pei and Han (2000), and later extended by Song and Qin (2005) into that can handle multiple convertible aggregation constraints. Then, Bonchi and Lucchese (2007) proposed data reduction technique and considered pushing tougher constraints in frequent pattern mining. Lee et al. (2006) proposed an approach to mine association rules with multiple aggregation constraints involving multi-dimensional attribute values.

2.3 Cardinality constraint

A cardinality constraint specifies requirement on the length of each pattern, which can also be referred to as the number of distinct items, or even the maximal number or minimal number of items per transactions. Such a requirement can be expressed in the form of $Card(S)\theta v$, $v \geq 0$. For example, $Card(S) \leq 7$ specifies that the cardinality (length) of each itemset S should be at most 7. Note that the cardinality constraint can also be anti-monotone or monotone (De Raedt and Zimmermann, 2007). An example situation for specifying this type of constraint is on the task of mining associative classification (Liu et al., 1998; Cho et al., 2005), which derive classification rules from generated frequent patterns, and shorter patterns result in classification rules with more generality and better understandability. Thus, the analyst can specify a constraint on limiting the length of patterns.

3 Mining frequent patterns with multiple constraints

3.1 Problem statement

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items, where each item is associated with a value attribute, such as cost, profit, or price. Let D be a transaction database consisting of a set of transactions, where a transaction $T = \langle tid, I_t \rangle$ is a set of items I_t with identifier tid and $I_t \subseteq I$. An itemset S, $S \subseteq I$, is contained in a transaction T if $S \subseteq I_t$. The support $sup(S)$ of an itemset S in a transaction database D is the fraction of transactions in D containing S. Given a support threshold ξ $(0 \leq \xi \leq 1)$, an itemset S is frequent provided $sup(S) \geq \xi$.

A constraint C is a predicate on the powerset of I, C: $\mathcal{P}(I) \rightarrow \{True, False\}$. An itemset S satisfies C if and only if $C(S) = True$. The complete set of itemsets satisfying constraint C is $SAT_C(I) = \{S \mid S \subseteq I \wedge C(S) = True\}$.

In this study, we consider three different kinds of constraints, including an item constraint C_B, a set of aggregation constraints SC and a cardinality constraint C_L. The problem of concern is to discover the set of itemsets F satisfying all constraints and minimum support threshold, i.e.,

$$F = \{S \mid S \in SAT_{C_B} \bigcap SAT_{SC} \bigcap SAT_{C_L} \wedge sup(S) \geq \xi\}. \quad (1)$$

Example 1: Consider the transaction database in Table 1 and the profit of each item in Table 2. Suppose that the user-specified constraints and support threshold are as shown in Table 3. Then, the set of frequent itemsets that satisfy the specified constraints are $\{pd, pdo, pdb, pdob\}$.

Table 1 A transaction database

TID	*List of items*
10	*b*, *c*, *d*, *o*, *p*
20	*b*, *c*, *e*
30	*b*, *d*, *o*, *p*, *r*
40	*a*, *b*, *d*, *p*
50	*b*, *d*, *e*, *o*, *p*

Table 2 Profit of each item in Table 1

Item	*Profit*
a	–20
b	30
c	0
d	10
e	–10
o	40
p	50
r	–30

Table 3 Settings of constraints

Constraint	*Value*
Aggregation	$avg(S) \geq 30$ and $sum(S) \geq 50$
Item	$p \wedge d$
Cardinality	$Card(S) \leq 7$
Support	Support threshold $\xi = 3$

3.2 Algorithm MCFPTree

In this subsection, we introduce the proposed algorithm, named Multi-Constrained Frequent Pattern Tree (MCFPTree) mining, which exploits the item constraint to construct the FP-tree and conditional FP-tree structures to discover satisfied frequent patterns. Compared with our previously proposed MCFP algorithm, MCFPTree is a relatively efficient method for mining constrained frequent patterns because by adopting the FP-tree structure MCFPTree does not have to generate candidate itemsets except the first phase for initial candidate generation.

Our MCFPTree algorithm is composed of five main phases:

1 initial candidate construction phase

2 database reduction and item counting phase

3 FP-Tree construction phase

4 frequent pattern generation phase

5 constraint checking phase.

In what follows, we will first detail each phase of MCFPTree, and then give an example to illustrate its execution.

3.2.1 Initial candidate construction phase

The initial candidate generation phase is the most critical step. MCFPTree exploits the item constraint directly to construct an initial set of candidate itemsets, with the intention to lessen the overhead in generating lots of intermediate candidates.

We consider two different cases:

1 the item constraint C_B does not contain any negative item

2 C_B contains some negative items.

Case 1: C_B constraint contains no negative item. In this case, we exploit the disjuncts of item constraint C_B to generate an initial set of candidate itemsets. For example, if $C_B = (a \wedge b) \vee (c \wedge d)$, then $K = \{ab, cd\}$.

Case 2: C_B contains some negative items. This case is far more complicated than the first case. The intuition is to avoid generating any candidate containing all positive items and part of the negative items in a disjunct D_i in C_B composed of negative items while not refrain the construction of cross-disjunct candidates that contain proper subsets of negative items in D_i and all positive items of another D_j.

In light of this, we first exploit the disjuncts of item constraint C_B to generate an initial set of candidate itemsets K that is composed of only positive items and move all negative items to a negative set N. Then, we generate the powerset of N, i.e., $\mathcal{P}(N)$. Finally, we perform a cross union between K and $\mathcal{P}(N)$, and prune those new initial candidate itemsets that contradict item constraint C_B. The detailed steps for accomplishing this procedure are described in Figure 1.

Figure 1 Procedure for generating initial itemsets from item constraint

Input: Item Constraint C_B.
Output: Set of initial candidate itemsets K.
Steps:

```
1.  K = N = K' = ∅;
2.  for each disjunct D_i of C_B do
3.      temp = ∅;
4.      for each item a_j in D_i do
5.          if a_j is negative item then
6.              N = N ∪ {a_j};
7.          else
8.              temp = temp ∪ {a_j};
9.          end if
10.     Insert temp into K;
11. end for
12. Generate the powerset of N, P(N);
13. for each ã ∈ K do
14.     for each ñ ∈ P(N) do
15.         temp = ã ∪ ñ;
16.         if temp can satisfy C_B then
17.             Insert temp into K';
18.     end for
19. end for
20. K= K';
```

Example 2: Consider an item constraint $C_B = (a \wedge b) \vee (b \wedge \sim d) \vee (c \wedge \sim p \wedge \sim r)$. First, we decompose the item constraint C_B to get the $K = \{ab, b, c\}$ and $N = \{d, p, r\}$. Then, we find out the powerset of N, $\mathcal{P}(N) = \{d, p, r, dp, dr, pr, dpr\}$. Finally, we perform a cross union between K and $\mathcal{P}(N)$, and prune those new initial candidates that do not satisfy C_B, resulting in the initial set of candidate itemsets $K = \{ab, b, c, abp, abr, abpr, bp, br, bpr, cd\}$.

3.2.2 Database reduction and item counting phase

In this phase, we scan the database to count the support of each item, and during which we also reduce and trim the transaction database according to the following rules: A transaction is pruned if it does not contain any initial candidate itemset. Finally, we prune all infrequent items.

3.2.3 FP-Tree construction phase

The task of this phase is to construct the FP-tree by scanning the reduced transaction database. The FP-tree structure and the steps for building it follow those used in FP-Growth (Han et al., 2000).

3.2.4 Frequent pattern generation phase

In this phase, we traverse the FP-tree to find out all frequent itemsets with support greater than or equal to ξ. Again, the approach used in FP-Growth is adopted. That is, we construct the conditional pattern base of each frequent 1-itemset, then construct its conditional FP-tree, and perform mining recursively on that tree to generate all of the frequent itemsets.

3.2.5 Constraint checking phase

In this phase, we check each of the frequent itemsets against the item constraint, aggregation constraints and the cardinality constraint to generate the set of satisfied frequent itemsets.

3.3 An example

Consider example 1 again. Here, we illustrate the process for executing MCFPTree on this example.

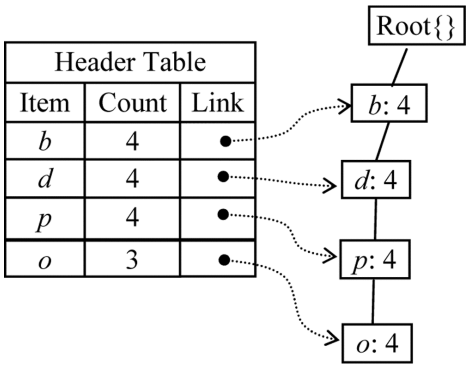
- *Phase 1*: Exploit the item constraint C_B to obtain initial candidate $\{pd\}$.
- *Phase 2*: Scan the transaction database to find all frequent 1-itemsets, obtaining $\{b, d, o, p\}$, and perform transaction trimming and reduction when appropriately. The resulting database is shown in Table 4.

Table 4 The reduced transaction database

TID	*List of items*
10	*b*, *c*, *d*, *o*, *p*
30	*b*, *d*, *o*, *p*, *r*
40	*a*, *b*, *d*, *p*
50	*b*, *d*, *e*, *o*, *p*

- *Phase 3*: Scan the reduced database to construct the FP-tree; the resulting FP-tree is shown in Figure 2.

Figure 2 Initial FP-tree



- *Phase 4*: This phase constructs the conditional FP-tree of each frequent 1-itemset to generate all frequent patterns, obtaining {*bdpo*, *dpo*, *po*, *bdp*, *dp*, *bd*}.
- *Phase 5*: Finally, all frequent itemsets are checked against all constraints. The resulting set of satisfied frequent patterns is shown in Table 5.

Table 5 The frequent itemsets that satisfy all sets of constraints

Itemset	*Support*	*sum*	*avg*	*Card*
dp	4	60	30	2
bdp	4	90	30	3
dpo	3	100	33.3	3
bdpo	3	130	32.5	3

4 Performance evaluations

In this section, we evaluate the performance of the proposed algorithm MCFPTree for mining frequent patterns with multiple constraints. A synthetic data set, T40I10D100K, generated by IBM generator (Agrawal and Srikant, 1994) and a real data set about traffic accident (Geurts et al., 2003) are used in this evaluation. Table 6 shows the parameter settings for generating T10I4D100K and the characteristics of accidents.

Table 6 Characteristics of T40I10D100K and accidents

Parameters		*T10I4D100K*	*Accidents*
$\|D\|$	Number of original transactions	100 K	341 K
$\|T\|$	Average size of transactions	10	33.8
$\|I\|$	Average size of frequent itemsets	4	–
$\|L\|$	Average size of maximal frequent itemsets	5	–
N	Number of items	1000	469

For comparison with our algorithms, we also implemented two methods: the MCFP algorithm and FP-Growth+ algorithm. Here, FP-Growth+ refers to the approach of applying the FP-Growth algorithm, followed by a post-processing of the frequent patterns. All experiments were performed on a two 1.8 GHz Intel Xeon CPUs ASUS server with 4 GB main memory and 450 GB hard disk running on Windows server 2003.

All comparisons were investigated from three different aspects, including support threshold, size of item constraints and number of aggregation constraints.

4.1 Performance on synthetic data

The effectiveness and efficiency of the proposed algorithm were first evaluated over the synthetic data set T10I4D100K. Table 7 shows the default constraint settings in this performance study.

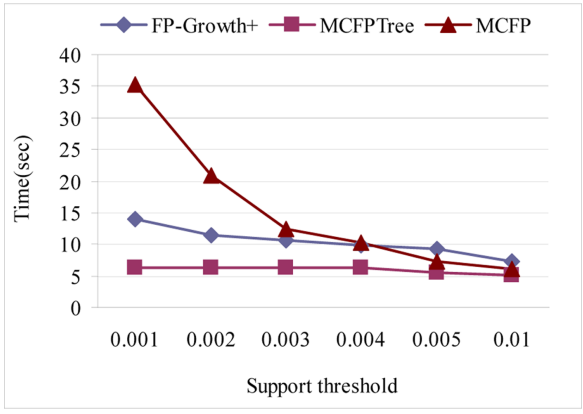
Table 7 Constraint settings in synthetic data set evaluation

Constraint	*Value*
Aggregation	$Avg(S) \geq 500$ and $sum(S) \geq 3000$
Item	$(900 \cap 851) \cup (701 \cap 800) \cup (800) \cup (700 \cap 9) \cup (750 \cap 9 \cap 3)$
Cardinality	$Card(S) \leq 7$

We first evaluate the performance from the aspect of various support thresholds, ranging from 0.1% to 1%. The other constraints consist of two aggregation constraints, one item constraint composed of five disjuncts, and a cardinality constraint, as shown in Table 7.

The result in Figure 3 demonstrates that MCFPTree is significantly faster than MCFP and FP-Growth+, especially on relative small support thresholds. FP-Growth+ outperforms MCFP on small support thresholds but is defeated on larger thresholds.

Figure 3 Performance comparison of FP-Growth+, MCFP and MCFPTree with varying support thresholds on T10I4D100K (see online version for colours)



Next, we inspect the effect of varying the size of item constraints. The support threshold is set to 0.002 and other constraint settings are the same as those in Table 7.

From the result in Figure 4, we observe that MCFPTree surpasses both FP-Growth+ and MCFP. The superiority over FP-Growth+ diminishes as # of disjuncts increases but against MCFP it enlarges. Note that the size of item constraints has almost no effect on algorithm FP-Growth because it does not utilise this constraint until the end of frequent itemset generation. Besides, the performance of MCFP is more affected by the number of disjuncts than MCFPTree. This is because MCFP is an Apriori-like algorithm (Agrawal and Srikant, 1994), whose computation needs to generate candidate itemsets. The more the number of disjuncts is, the more the number of candidates will be generated.

Finally, we consider the effect of varying the number of aggregation constraints, which is set from 1 to 6. Other parameter settings in this experiment are the same as before.

The result in Figure 5 shows that the performance of MCFP is heavily affected by the number of aggregation constraints; the larger the number of aggregation constraints is, the worst the MCFP performs. On the contrary, our MCFPTree algorithm is not affected

by varying number of aggregation constraints. The reason is that the aggregation constraints are utilised only in the post-processing phase, whose cost is negligible.

Figure 4 Performance comparison of FP-Growth+, MCFP and MCFPTree with varying number of disjuncts on T10I4D100K (see online version for colours)

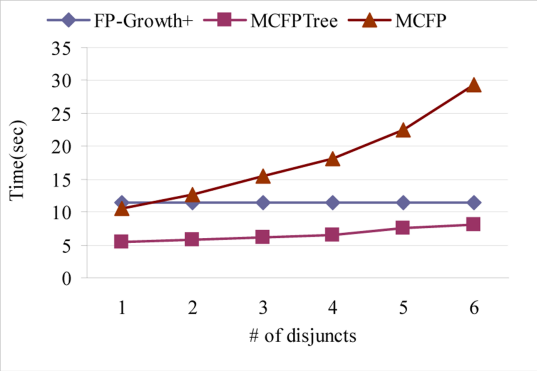
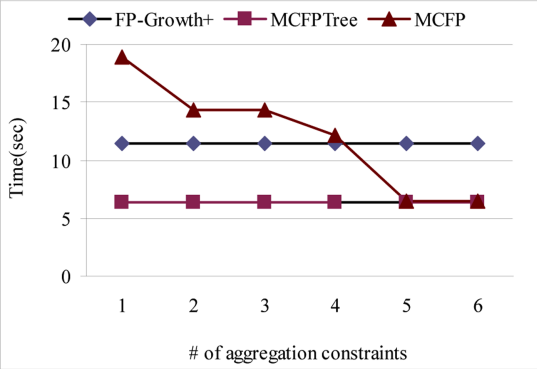


Figure 5 Performance comparison of FP-Growth+, MCFP and MCFPTree with varying number of aggregation constraints on T10I4D100K (see online version for colours)



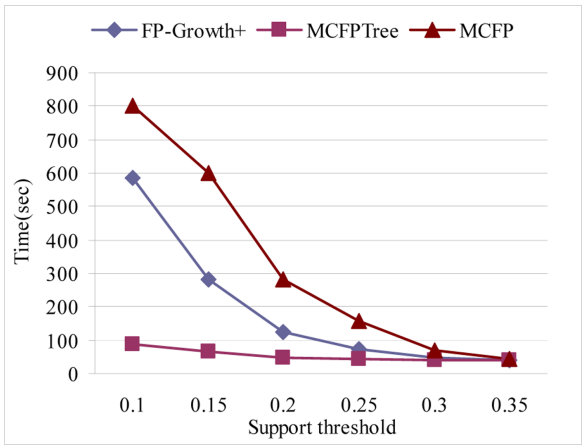
4.2 Performance on real data

First, we consider the effect of varying support threshold, ranging from 10% to 35%. The other constraints consist of two aggregation constraints, one item constraint composed of five disjuncts, and a cardinality constraint. Table 8 shows the detailed settings. The result is depicted in Figure 6, which is very similar to that shown in Figure 3 except that FP-Growth+ outperforms MCFP in all cases of support thresholds.

Table 8 Constraint settings in real data set evaluation

Constraint	*Value*
Aggregation	$avg(S) \geq 35$ and $sum(S) \geq 150$
Item	$(183 \wedge 34) \vee (59 \wedge 24) \vee 82 \vee (12 \wedge 36) \vee 173$
Cardinality	$Card(S) \leq 7$

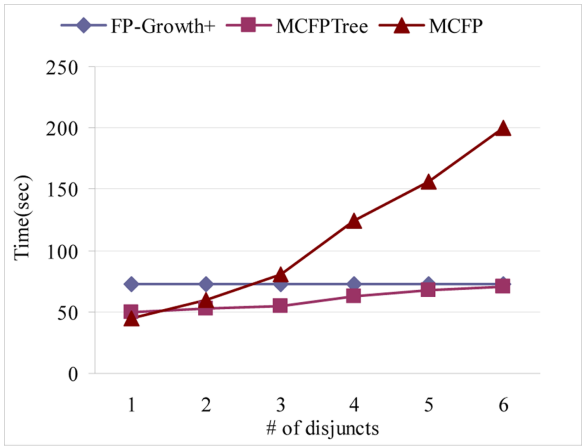
Figure 6 Performance comparison of FP-Growth+, MCFP and MCFPTree with varying support thresholds on accidents (see online version for colours)



Next, we consider the effect of varying the size of item constraint, which is measured by the number of disjuncts ranging from 1 to 6. The other parameter settings are the same as before except that the support threshold is set to 0.25.

Again, one can observe that the result in Figure 7 is very similar to that shown in Figure 4. But, MCFP can beat FP-Growth+ and MCFPTree when the number of disjuncts is less than 3 and 2, respectively. This is because smaller number of disjuncts means less number of candidate itemsets needed to be generated during the computation of MCFP.

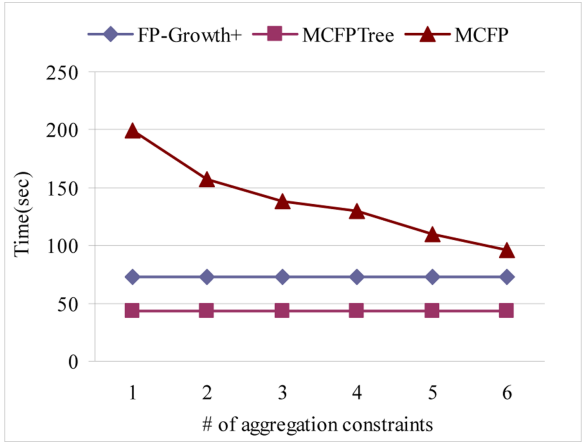
Figure 7 Performance comparison of FP-Growth+, MCFP and MCFPTree with varying number of disjuncts on accidents (see online version for colours)



Finally, we consider the effect of varying the number of aggregation constraints, which is set from 1 to 6. The other parameter settings in this experiment are the same as before.

As the result shown in Figure 8, the number of aggregation constraints does not affect the performance of FP-Growth+ and MCFPTree because this constraint is inspected only in the post-processing phase. However, MCFP is heavily affected by the number of aggregation constraints; the more the number of constraints, the less its performance.

Figure 8 Performance comparison of FP-Growth+, MCFP and MCFPTree with varying number of aggregation constraints on accidents (see online version for colours)



5 Conclusions

Recent work has highlighted the essence of allowing user-specified constraints into the model of association rules to facilitate an online, interactive mining environment of association rules. The key for realising such a mining system is the design of an efficient frequent pattern-mining algorithm that takes account of all user-specified constraints.

In this paper, we have proposed a new algorithm MCFPTree for discovering patterns that satisfy three types of user-specified constraints, including item constraint, aggregation constraint and cardinality constraint. Experimental results show that our algorithm MCFPTree is significantly faster than our previously proposed Apriori-like algorithm MCFP and also outperform an intuitive approach, post-processing the frequent patterns generated by the leading algorithm FP-Growth against user-specified constraints.

In real life, a transaction table may contain multiple attributes and users sometimes may need to select different attributes from the multi-dimensional data sets, i.e., data warehouse (Han and Kamber, 2004; Inmon and Kelley, 1993), to mine multi-dimensional association rules (Han et al., 1997; Perng et al., 2001; Tjioe and Taniar, 2005). So, a promising avenue of extending our work is to discover constrained rules from multi-dimensional data rather than single-dimensional data.

Acknowledgement

This work is partially supported by National Science Council of Taiwan with grant No. 95-2221-E-390-024.

References

Agrawal, R. and Srikant, R. (1994) 'Fast algorithms for mining association rules in large databases', *Proc. 20th International Conference on Very Large Data Bases*, Santiago de Chile, Chile, pp.487–499.

Bayardo Jr., R.J., Agrawal, R. and Gunopulos, D. (2000) 'Constraint-based rule mining in large, dense databases', *Data Mining and Knowledge Discovery*, Vol. 4, pp.217–240.

Bonchi, F. and Lucchese, C. (2007) 'Extending the state-of-the-art of constraint-based pattern discovery', *Data and Knowledge Engineering*, Vol. 60, No. 2, pp.377–399.

Bonchi, F., Giannotti, F., Mazzanti, A. and Pedreschi, D. (2003) 'ExAMiner: Optimized level-wise frequent pattern mining with monotone constraint', *Proc. IEEE International Conference on Data Mining*, Melbourne, Florida, pp.11–18.

Ceglar, A. and Roddick, J.F. (2007) 'GAM: A guidance enabled association mining environment', *International Journal of Business Intelligence and Data Mining*, Vol. 2, No. 1, pp.3–28.

Cho, M., Pei, J., Wang, H. and Wang, W. (2005) 'Preference-based frequent pattern mining', *International Journal of Data Warehousing and Mining*, Vol. 1, No. 4, pp.56–77.

De Raedt, L. and Zimmermann, A. (2007) 'Constraint-based pattern set mining', *Proc. SIAM International Conference on Data Mining*, Minneapolis, Minnesota, pp.237–248.

Geurts, K., Wets, G., Brijs, T. and Vanhoof, K. (2003) 'Profiling high frequency accident locations using association rules', *Proc. 82nd Annual Transportation Research Board Annual Meeting*, Washington, DC, pp.123–130.

Grahne, G., Lakshmanan, L.V.S. and Wang, X. (2000) 'Efficient mining of constrained correlated sets', *Proc. 16th International Conference on Data Engineering*, San Diego, California, pp.512–521.

Han, J. and Kamber, M. (2004) *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, California.

Han, J., Kamber, M. and Chiang, J. (1997) 'Metarule-guided mining of multi-dimensional association rules using data cubes', *Proc. 3rd International Conference on Knowledge Discovery and Data Mining*, Newport Beach, California, pp.207–210.

Han, J., Pei, J. and Yin, Y. (2000) 'Mining frequent patterns without candidate generation', *Proc. 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, Texas, pp.1–12.

Inmon, W. and Kelley, C. (1993) *Developing the Data Warehouse*, QED Publishing Group, Boston, Massachussetts.

Lee, A.J.T., Lin, W.C. and Wang, C.S. (2006) 'Mining association rules with multi-dimensional constraints', *Journal of Systems and Software*, Vol. 79, No. 1, pp.79–92.

Lin, W.Y., Huang, K.W., Li, H.Y. and Jiang, C.L. (2008) 'Mining frequent patterns with item, aggregation, and cardinality constraints', *Proc. 3rd International Conference on Innovative Computing, Information and Control*, Dalian, China, pp.325–328.

Liu, B., Hsu, W. and Ma, Y. (1998) 'Integrating classification and association rule mining', *Proc. International Conference on Knowledge Discovery and Data Mining*, New York, pp.80–86.

Lu, N., Zhe, W., Zhou, C.G. and Zhou, J.Z. (2005) 'Research on association rules mining algorithm with item constraints', *Proc. 2005 International Conference on Cyberworlds*, Singapore, pp.325–329.

Ng, R.T., Lakshmanan, L.V.S., Han, J. and Pang, A. (1998) 'Exploratory mining and pruning optimizations of constrained association rules', *Proc. 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, pp.13–24.

Pei, J. and Han, J. (2000) 'Can we push more constraints into frequent pattern mining?', *Proc. 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, Massachusetts, pp.350–354.

Pei, J., Han, J. and Lakshmanan, L.V.S. (2004) 'Pushing convertible constraints in frequent itemset mining', *Data Mining and Knowledge Discovery*, Vol. 8, No. 3, pp.227–252.

Perng, C.S., Wang, H., Ma, S. and Hellerstein, J.L. (2001) 'Farm: A framework for exploring mining spaces with multiple attributes', *Proc. 2001 IEEE International Conference on Data Mining*, San Jose, California, pp.449–456.

Song, B.L. and Qin, Z. (2005) 'Efficient mining for frequent itemsets with multiple convertible constraints', *Proc. of 4th International Conference on Machine Learning and Cybernetics*, Guangzhou, China, pp.1503–1508.

Srikant, R., Vu, Q. and Agrawal, R. (1997) 'Mining association rules with item constraints', *Proc. of 3rd International Conference on Knowledge Discovery and Data Mining*, Newport Beach, California, pp.67–73.

Tjioe, H.C. and Taniar, D. (2005) 'Mining association rules in data warehouses', *International Journal of Data Warehousing and Mining*, Vol. 1, No. 3, pp.28–62.

Zaki, M.J. (2000) 'Scalable algorithms for association mining', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 3, pp.372–390.