ISSN - 2277 - 0593

Journal of Natural Sciences, Engineering and Technology

AN IMPROVED TECHNIQUE FOR MULTI-DIMENSIONAL CONSTRAINED GRADIENT MINING

O.J. ELUGBADEBO*[1], A.S. SODIYA[2], O. FOLORUNSO[2]

[1]Federal College of Education, Osiele, Abeokuta, Ogun State, Nigeria.
[2]University of Agriculture, Abeokuta, Ogun State, Nigeria.
folorunsolusegun@yahoo.com, sinaronke@yahoo.co.uk
***Corresponding author:** jossyday@yahoo.com

ABSTRACT

Multi-dimensional Constrained Gradient Mining, which is an aspect of data mining, is based on mining constrained frequent gradient pattern pairs with significant difference in their measures in transactional database. Top-k Fp-growth with Gradient Pruning and Top-k Fp-growth with No Gradient Pruning were the two algorithms used for Multi-dimensional Constrained Gradient Mining in previous studies. However, these algorithms have their shortcomings. The first requires construction of Fp-tree before searching through the database and the second algorithm requires searching of database twice in finding frequent pattern pairs. These cause the problems of using large amount of time and memory space, which retrogressively make mining of database cumbersome. Based on this anomaly, a new algorithm that combines Top-k Fp-growth with Gradient pruning and Top-k Fp-growth with No Gradient pruning is designed to eliminate these drawbacks. The new algorithm called Top-K Fp-growth with support Gradient pruning (SUPGRAP) employs the method of scanning the database once, by searching for the node and all the descendant of the node of every task at each level. The idea is to form projected Multidimensional Database and then find the Multidimensional patterns within the projected databases. The evaluation of the new algorithm shows significant improvement in terms of time and space required over the existing algorithms.

Keywords: Data mining, Association rules, Frequent itemsets, Multidimensional, Constraints, Gradient.

INTRODUCTION

Data mining, also called Knowledge Discovery, is a multi-disciplinary field including database, artificial intelligence (especially machine learning) and statistics. In general, data mining is defined as the process of automatic extraction of implicit, novel, useful and understandable patterns in large databases (Guozhu *et al.,* 2001). The task of data mining, which is finding hidden and predictive information from large amount of data collected through daily operation, has been a major concern of today's business world. Data mining pursues a systematic approach in data analysis and helps businesses leverage the knowledge hidden in their own data by adjusting their business strategies accordingly.

Analyzing large amount of data collected through daily operation is a non-trivial task, let alone analyzing data from a multi-dimensional point of view. Thus, multi-dimensional data analysis has been the focus of recent research in the field of data mining. It pursues a systematic approach in data analysis and helps businesses leverage the knowledge hidden in their own data by adjusting their business strategies accordingly (Tomasz *et al.,* 2002). Also, multi-

dimensional constraint gradient in Transactional Database is basically used to correct the problem encountered in transactional database environment, most especially in finding pairs of frequent patterns with significant difference in their measures (Joyce Man, 2001).

Nowadays, data are usually collected in tables stored in relational database systems. Each table has a schema. An attribute in the schema can be treated as a dimension. For example, a "Customer Category" attribute can be treated as a dimension with its own set of unique values and possibly a concept hierarchy for organizing values into different levels of granularity. Data having multiple dimensions can be arranged into a lattice of data cubes and viewed in a multi-dimension way. Similarly, in a transaction database, each tuple records the items (or products) bought in a transaction. Often it records other auxiliary information about a transaction such as the time the transaction happened and the location the transaction took place in. Attributes representing this auxiliary information can be treated as dimensions. In this case we can view a transaction as multi-dimensional. (Guozhu *et al.,* 2001).

Constraints are user-specified conditions or restrictions that an answer must satisfy (Agrawal *et al.,* 1993). There are restrictions indicating users' interest in seeing what to report. Otherwise, returning a large answer set might overwhelm the users and reporting unusable answers is a waste of computation time.

In this work, we adopt the "gradient" definition in a previous related work (Joyce Man, 2001) such that its magnitude, expressed as a ratio of measures, indicates the change between two multi-dimensional entities (patterns in transaction database).

The main objectives of this work are the technical issues of frequent pattern mining, identifying and analyzing algorithms for mining Multi-dimensional Constrained Gradient in Transactions database, developing an improved technique (SUPGRAP) that will combine two different pruning strategies: (support pruning and gradient pruning) as they are currently treated independently, testing and evaluating the new algorithm.

The rest of this paper is organized as follows: section two presents review of existing literature on mining association rule in transactional databases. Section three discusses the improved algorithm. The implementation and evaluation were described in section four. In section five, suggested future work were highlighted and the work is concluded in section six.

REVIEW OF LITERATURE

Agrawal *et al.* (1993), mining association rules in databases has been a topic of choice among researchers. Since then, tremendous results have been recorded in developing different algorithms to mine frequent itemsets and association rules.

Frequent Pattern and Association Rule Mining

Frequent pattern mining is one of the achieve research themes in data mining. It is a process of applying data mining algorithm to find frequent patterns in a large database. The frequent pattern is a set of items that occurs frequently in a database (Ibrahim, 2004).

Association rule mining discovers interesting relationships among items in a given data set

(Guozhu *et al.,* 2004). The basic concept can be illustrated under the context of Market Basket Analysis. Imagine you are a retail store manager who wants to analyze your customers' buying behavior. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items or products in your store. Let T be a set of task-relevant transactions where each $t \subseteq T$ is a set of items such that $t \subseteq I$. Each t has a unique transaction identifier TID. Let A and B be itemsets, i.e. $\subseteq$ sets of items. (An item set is also $\subseteq$ called a *pattern*.) An item set containing k items is called a *k-itemset*. A transaction t contains an itemset A if $\subseteq$ and only if $A \subseteq t$ An association rule $\subseteq$ is of the form $A => B$ with support s and confidence c $\subseteq$ where $A \subseteq I$, $B \subseteq I$ and $A \cap B = \emptyset$, $\subset$ where A and B are itemsets. *Support s* is defined $\subset$ as the $\subset$ percentage or absolute number of transactions in T that contain $A \cup B$, i.e. probability $P(A \cup B)$. Of all the transactions in T which contain A, *confidence c* indicates the percentage or absolute number of these transactions which also contain B, that is the conditional probability $P(B \mid A)$. To determine the interestingness of an association rule, minimum support threshold (min_sup) and minimum confidence threshold (min_conf) are used. An interesting or strong association rule must satisfy both minimum thresholds. Association rule mining is a two-step process Wang *et al.* (2006).

- Find all frequent itemsets (i.e. itemsets which occur at least in *min_sup* number of transactions). These frequent itemsets are also called *large* itemsets (or patterns),
- Generate strong association rules from these frequent itemsets.

In this two-step process, the first step is the dominant time-consuming step. Thus, most research effort has been dedicated into finding efficient algorithms to mine frequent itemsets (or patterns). Below, we will discuss two influential algorithms, *Apriori* and *FP-growth*, in finding all frequent itemsets. For details in generating association rules from the frequent itemsets, please refer to (Agrawal *et al.,* 1993).

Agrawal *et al.* (1993), *Apriori* has become the classic algorithm for finding frequent itemsets. With an iterative *level-wise* search approach, it uses frequent k-itemsets to explore frequent (k+1)-candidate itemsets through join and prune steps. In general, first the set of frequent 1-itemsets, L_1, is found. L_1 is used to find the set of frequent 2-itemsets L_2, which is used to find L_3, and so on. This algorithm terminates when no further frequent k-itemsets L_k can be found. In the process of finding each L_k, one full scan of the database is required. To improve the efficiency of this iterative level-wise approach, the *Apriori heuristic* is used for pruning to narrow down the search space. The anti-monotonic Apriori heuristic (Agrawal et al., 1994) states that *if any length k pattern (itemset) is not frequent, its length (k+1) super-pattern can never be frequent*. This is based on the observation that if an itemset I do not satisfy the minimum support threshold, then I is not frequent and any superset of I will also be infrequent.

Ansari and Sadreddini (2009), in their research titled "An Efficient Approach to Mining Frequent Itemsets on Data Streams" which is also an aspect of this work used the approach "Sructured Frequent Itemsets on Data Streams (SFIDS) algorithm" for the implementation of Frequent Itemsets on Data Streams (FIDS) whereby the most frequent itemsets on data streams were derived

using one search method and it is one – dimensional approach while our own approach tailored on multi – dimensional approach because it involves two phases embedded in one algorithm called Support Gradient Prunning (SUPGRAP). The first phase involves searching for the most frequent itemsets in a transactional database and the second involves the extraction of the most profitable frequent itemsets from the result of the first phase.

More often, there are some challenges inherent in frequent pattern mining. There is a challenge of how to reduce the multiple scanning of the transaction database so as to improve system's performance. There are challenges of how to reduce the massive number of candidates generated during the processes and to totally eliminate the tedious workload of support counting of the generated candidates. These ideas lead to diverse options enhancements of the Apriori algorithm (Nehinbe, 2004)

As described above, the disadvantages of Apriori algorithm are its expensive candidate generation process that results in a huge number of candidates and the requirement of scanning the entire database once at each level. Focusing on this weakness, *FP -growth* (Jiawei *et al.,* 2000) minimizes the candidate generation process to only those most likely to be frequent and adopts a compact prefix tree data structure, *FP-tree*, to avoid repetitive scanning of the database. In this section, we examine FP-growth algorithm with an example since it is used in multi-dimensional constrained gradient mining in transaction databases (refer to Section 3). The following example is taken from (Jiawei *et al.,* 2000) to illustrate the working of FP-growth. Assume transaction database T is shown in Table 1 and the minimum support threshold is 3.

First, perform one scan of the transaction database T to find all frequent 1-itemsets. During scanning, frequency count for each item is tracked and compared with the minimum support threshold to determine if it is frequent. Only frequent items are of interest to us. Then we sort these frequent 1-itemsets in descending order of their frequency counts: <(f:4), (c:4), (a:3), (b:3), (m:3), (p:3)> (The number after ":" denotes the frequency count of an itemset. Then items in each transaction are sorted in this same order before being inserted into the FP-tree, as shown in the column "(Ordered) Frequent Items". This facilitates maximal sharing of nodes since items with higher frequencies appear closer to the root. Thus, the size of FP-tree created will be small. A second scan of T is required to create this compact FP-tree structure. After reading the first transaction, we insert each item into the tree and construct the first branch: <(f:1), (c:1), (a:1), (m:1), (p:1)>. After reading the second transaction in T <f, c, a, b, m>, we find that it shares a common prefix path <f, c, a> with the existing branch <f, c, a, m, p> in the tree. Therefore, the count for each node in the prefix path is incremented by 1 and a new node (b:1) is created and linked as the child of node (a:2). Another new node (m:1) is created and linked as the child of (b:1). Figure 1 shows the FP-tree after insertion of two transactions.

Each subsequent transaction is scanned and inserted into the FP-tree. If a common prefix path already exists in the tree, the count in these common nodes are incremented. Otherwise, new nodes are created and inserted into the FP-tree to accomplish insertion of a new transaction. Figure 2 shows the complete FP-tree for database T.

Table 1: Transaction table, with ordered frequent items

Trans ID	Items Bought	(Ordered) Frequent Items
1	f, a, c, d, g, i, m, p	f, c, a, m, p
2	a, b, c, f, i, m, o	f, c, a, b, m
3	b, f, h, j, o	f, b
4	b, c, k, s, p	c, b, p
5	a, f, c, e, i, p, m, n	f, c, a, m, p

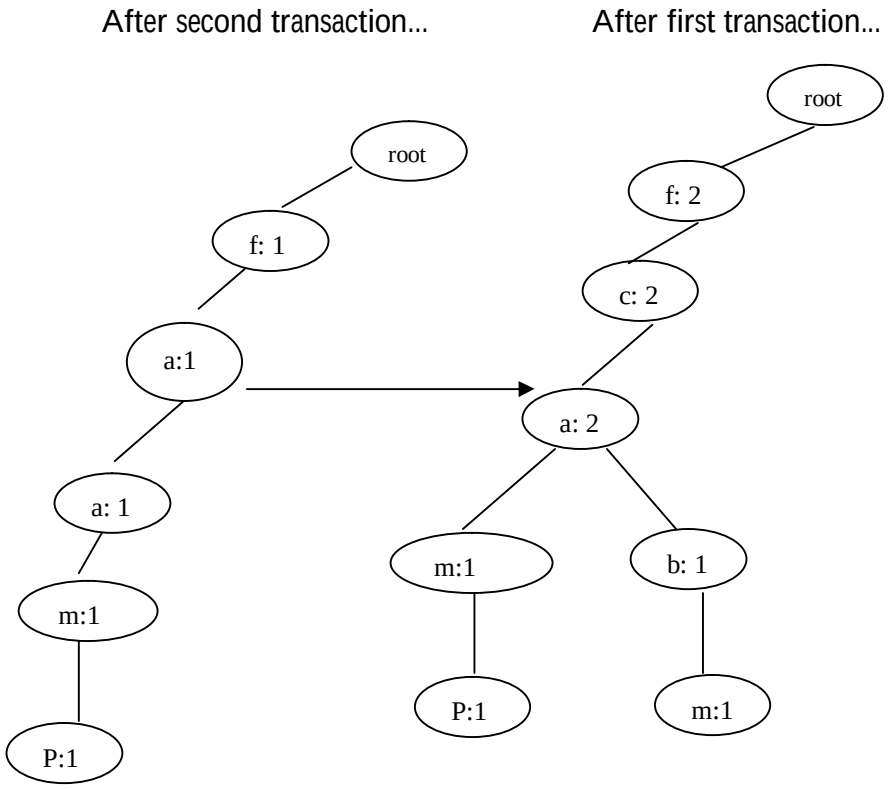

Figure 1: FP-tree after insertion of two transactions

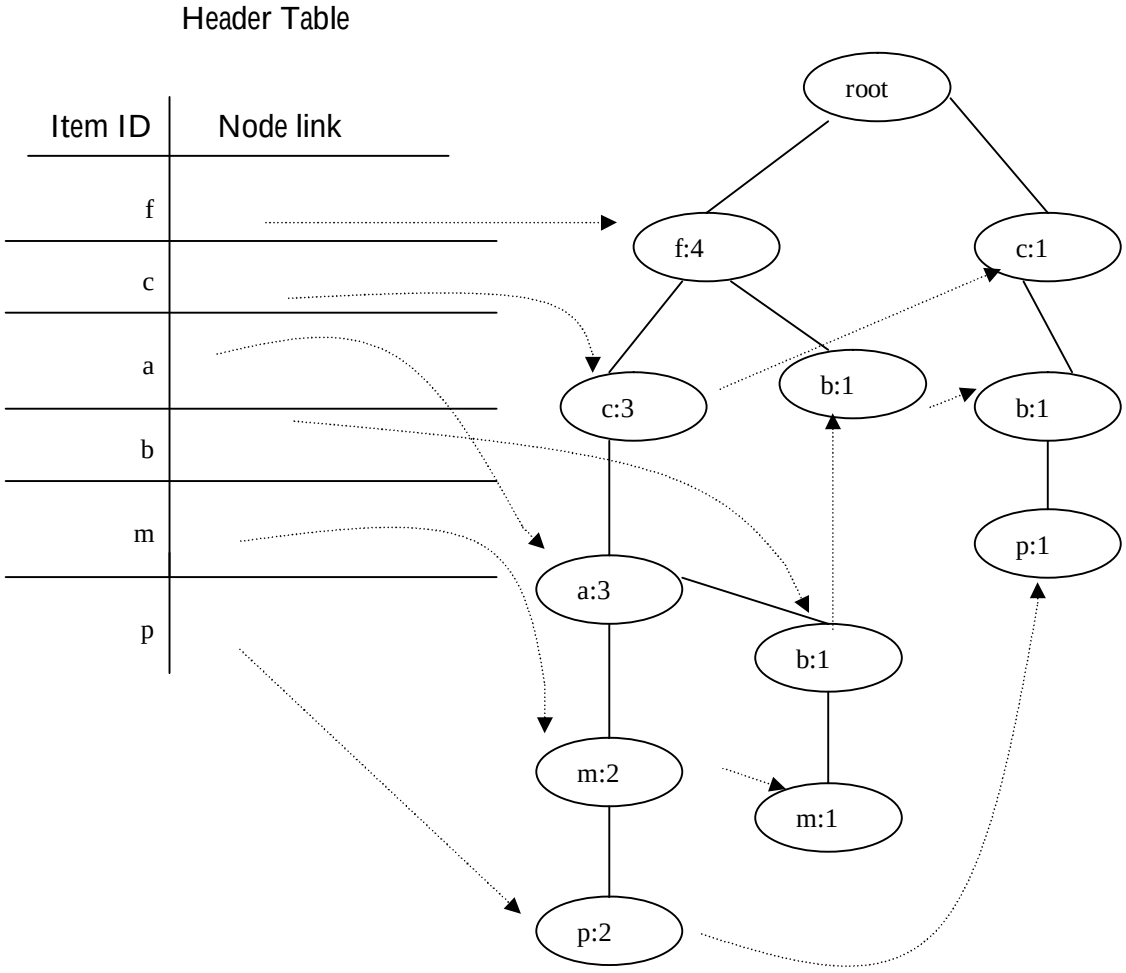

Figure 2: Sample FP-tree for database T

To facilitate tree traversal, a *frequent-item header table* is built during FP-tree construction. Each entry consists of an item ID and the head of a node-link sequence to keep track of the first occurrence of each item in the tree. Items are arranged in descending frequency order in the header table. Each node in the tree has a link to the next occurrence of the same item. By traversing this sequence of node-links, one can visit all occurrences of the same item in the tree. FP -growth mining algorithm starts with the item having the least frequency count in the header table. In this example, we start with item <p> as the initial *suffix* pattern. Immediately from the header table, we generate the frequent itemset <p:3>. By traversing the node links for item <p>, we extract its prefix patterns, namely <f:2, c:2, a:2, m:2> and < c:1, b:1>. These prefix patterns form <p>'s *conditional pattern base* (i.e. the sub-pattern base under the condition of p's existence). Recursively, a *conditional Fp-tree* is constructed for the suffix pattern, <p>, based

on the conditional pattern base. This conditional FP-tree only has one branch (c:3) because only c is frequent by satisfying the minimum support threshold. Mining of <p>'s conditional FP-tree produces frequent 2-itemset (cp:3). Mining based on suffix pattern <p> terminates. We move on to the next suffix pattern <m:3> in the header table. The conditional pattern base for suffix pattern <m> includes prefix paths <f:2, c:2, a:2> and <f:1, c:1, a:1, b:1>. Recursive mining of this conditional pattern base for <m> generates the frequent itemsets <am:3>, <cm:3>, <fm:3>, <cam:3>, <fam:3>, <fcam:3>, and <fcm:3>. In short, concatenating each frequent item in the conditional pattern base with the suffix pattern produces frequent itemsets of increasing lengths. Once mining of a suffix pattern has finished, we move on to the next more frequent item in the header table until the item with the highest frequency has been mined.

Mining starts from the item of least frequency in the header table because the conditional FP-tree generated for this least frequent item will still be compact since higher frequency items allow more sharing. All frequent itemsets with the least frequent item, as the suffix pattern will be generated. Subsequently, conditional FP-trees for higher frequency items can ignore lower frequency items so that subsequent conditional FP-trees are smaller in size. The compact FP-tree structure eliminates the need to have repetitive database scans in order to generate candidate itemsets as in Apriori since all frequent itemsets must exist as a branch in the FP-tree. As mining proceeds in this FP-tree, we only need to check if an itemset's count meets the minimum support threshold.

In summary, FP-growth algorithm only scans the whole database twice and mines on the compact data structure, FP-tree, to find all frequent itemsets without generating all possible candidate sets. Thus, FP-growth algorithm is a fast algorithm for mining frequent itemsets for association rule mining.

METHODOLOGY

Problem Definition

Informally, we can state the problem as "Given a frequent promotion pattern (f_p) containing a set of promotion item(s) P and average profit of these promotion items in f_p AvgProfitP(f_p), we want to find the set of all frequent gradient patterns such that each frequent gradient pattern f_g contains a set of items (including all promotion item(s) in P) and an AvgProfitP(f_g) that is at least x% higher than that of f_p's."(plural of f_p)

Simple frequent pattern mining finds the complete set of frequent patterns F in a database. A frequent pattern f Î F is considered frequent (or significant) if it satisfies a significance constraint C_{sig} (or support threshold). A frequent promotion pattern (f_p) contains a set of promotion items defined by a k-conjunct of items, i.e. a promotion constraint C_{pro}. Promotion constraint C_{pro} has the format of k-conjunct items = { id1 $\wedge$ $\wedge$ idk | k ≥ 1, idj ϵ I} where I is a set of all unique items in T. Referring to our earlier example, C_{pro} = {TV}. This frequent promotion pattern f_p is used as the basis for comparison with other potential frequent gradient patterns f_g, which are super patterns of f_p (i.e. $f_p \subset f_g$). A gradient constraint Cgrad is expressed as a gradient function on frequent gradient pattern f_g and frequent promotion pattern f_p. It has the form

$$C_{grad}(f_g, f_p) \; ^o g(f_g, f_p) \; \theta \; v$$

where g is a gradient function, θ is a symbol

of <, >,≤, ≥, etc. and *v* is a constant value. A frequent gradient pattern f_g must

- be frequent (or significant) based on significance constraint C_{sig},
- contains all promotion items defined in promotion constraint C_{pro}, and
- be $C_{grad}(f_g, f_p)$ = true.

We limit our study of gradient constraint to the ratio of f_g's and f_p's measures such that

$$Cgrad(f_g, f_p) = \frac{m(f_g)}{m(f_p)} \theta v$$

where m(f) is an arbitrary measure for a frequent pattern f. In this study, we focus on complex measures, e.g. m = AvgProfit of promotion items P = AvgProfit (P). (Joyce *et al.,* 2001)

Let T be a multi-dimensional transaction database with schema S where S contains two non-overlapping sets of attributes: dimensional attributes D and measure attributes M, and a set of items bought together in a transaction. Let pattern be a set of items occurring together. Let fg be a gradient pattern and fp a promotion pattern. Given a significance constraint Csig, a promotion constraint Cpro and a gradient constraint Cgrad, find all valid gradient-promotion pattern pairs (fg, fp) in T such that

- the set of promotion items p ϵP is defined by Cpro,
- fp is a frequent pattern containing all p Î P,
- fg and fp are frequent (or significant) patterns (i.e. satisfy Csig),
- fg must be a superpattern of fp (i.e. fp ⊂ fg),
- fg's measure, m(fg), must satisfy Cgrad, and
- m is a complex measure like AvgProfit of promotion items.

In subsequent sections, we first reason our choice on applying FP-tree/FP-growth to this problem instead of H-tree/H-cubing. Afterwards, we focus on each of the two steps involved in mining multi-dimensional constrained gradient patterns:

- Construction of Top-k FP-tree data structure;
- Mining results on Top-k FP-tree with Top-k FP-growth.

Top-k FP-growth with No Gradient Pruning algorithm

Since we use Top-k FP-tree as the data structure, one simple method to solve multi-dimensional constrained gradient mining in transaction database is to adopt FP-growth algorithm on Top-k FP-tree (Joyce Man, 2001). Given promotion constraint C_{pro}, we find a set of promotion item(s) p ϵ P, calculate AvgProfit(P) in transaction database and construct corresponding Top-k FP-tree. We apply FP-growth algorithm using significance constraint C_{sig} as its support threshold to find a set of frequent patterns from Top-k FP-tree. This set of frequent patterns resulting from pure support pruning is a superset of the set of frequent gradient patterns. Thus, an iterative post-processing step is required to filter out patterns that do not satisfy the derived gradient pattern threshold C_{gpat}. We call this method *Top-k FP-growth with No Gradient Pruning (TopkFpNoGP).*

Top-k FP-growth with No Gradient Pruning

Algorithm (Top-k FP-growth with No Gradient Pruning)

Input: A multi-dimensional transaction database T, a significance constraint C_{sig}, a pro-

motion constraint C_{pro}, a gradient constraint C_{grad}, and size of each bin.

Output: The complete set of frequent gradient patterns that can form valid gradient promotion patterns satisfying all constraints with frequent promotion pattern f_p that contains the set of promotion items P.

Method

1) Construct a Top-k FP-tree as described in section **"Considerations for the new algorithm"**

2) Apply FP-growth (as described in Chapter 2) on Top-k FP-tree using C_{sig} as support threshold.

3) Derive C_{gpat} using C_{grad} and AvgProfit(P) in T.

4) FOR EACH frequent pattern f found DO // post-processing

{

a) If frequent pattern f's AvgProfit(P) passes C_{gpat} threshold

i) Report as frequent gradient pattern f_g

}

However, this algorithm searches through fp tree which means it search from node to node before considering its descendants one after the other, because of this much of the user's time is consumed. As a result of this drawback, Joyce Man, (2001) developed and implemented another algorithm called Top-k Fp-growth with Gradient Pruning (TOPFPGP) as described below:

Algorithm (Top-k FP-growth with Gradient Pruning)

Input: A multi-dimensional transaction database T, a significance constraint C_{sig},, a promotion constraint C_{pro}, a gradient constraint C_{grad}, and size of each bin

***Output*:** The complete set of frequent gradient-promotion pattern pairs that satisfy the three constraints.

Method

1) Scan T once to get projected database on promotion item(s) P (based on C_{pro}) T' and calculate AvgProfit(P) in T. Remove promotion item(s) from each transaction t' Î T'. Schema for T' is mapped to:

T' (# items in transaction, itemID, …, itemID, profit(P)) where an itemID can be a unique dimension value or an item bought in t'.

2) Scan T' once to get frequent large-1 items using C_{sig}.

3) If (number of large-1 items > 0)

a) Derive gradient pattern threshold C_{gpat} from C_{grad} and AvgProfit(P) in T.

b) Calculate the number of bins required using average and minimum profit of P and user-specified bin size.

c) Scan each transaction t' Î T' second time to construct Top-k FP-tree.

d) Call Top-k-FP-growth(top-k FP-tree, null) using C_{sig} and C_{gpat}…..(ii)

Example (Top-k FP-growth algorithm)

Based on Top-k FP-tree constructed for Table 1, we know AvgProfit(C_g) $\geq$ C_{grad} x AvgProfit(P) = 110, and C_{sig} = 3. Let k = C_{sig} = 3. Top-k FP-growth algorithm starts mining from the last item in header table, i.e. p. Pattern (p:3) is frequent with top-3 average = actual average = (200+80+60)/3 = 113.3 $\geq 110 = C_{gpat}$. So (p:3, 113.3) is a frequent gradient pattern which can form a gradient-promotion pattern pair with promotion items P. Since top-3 average of p passes C_{gpat}, we recursively construct conditional FP -tree for p and continue mining base on prefix pattern (p). As we can see, (cp: 3) is the only frequent pattern. Its top-3 average = actual average = 113.3 $\geq 110 = C_{gpat}$ so (cp: 3, 113.3) is also a frequent gradient pattern. Let's skip to see how we mine Top-k FP-tree

in Figure 3 from item f in header table. Pattern (f:4) is frequent with top-3 average = (200+100+60)/3 = 120 $\geq$ 110 = C_{gpat}. However, (f:4)'s actual average = (200+100+60+60)/4 = 105 $\geq$ 110 = C_{gpat} so even though (f:4) is frequent, it is not a frequent gradient pattern. Since (f:4) passes top-3 average, we continue mining its conditional FP-tree with prefix path (f).

Mining from the last item in Header Table f, pattern (cf: 3) is frequent with top-3 average = actual average = (200+100+60)/3 = 120 $\geq$ 110 = C_{gpat} so (cf: 3, 120) is a frequent gradient pattern. Recursively mining the conditional FP-tree for prefix path (c_f), we find pattern (acf: 3) is a frequent pattern with top-3 average actual average = (200+100+60)/3 = 120 $\geq$ 110 = C_{gpat} so (acf: 3, 120) is a frequent gradient pattern also. Next, we move on to item b in Header Table f, pattern (bf: 2) is not frequent so terminate searching for patterns with prefix path (b_f). We move on to item a in Header Table f, pattern (af: 3) is frequent and top-3 average = actual average = 120 $\geq$110 = C_{gpat} so (af: 3, 120) is a frequent gradient pattern.

The final answer of the complete set of frequent gradient patterns from Figure 1 is listed in Table 2.

According to Joyce Man (2001), in his final analyzes, he found that the second algorithm which was based on scanning of the database in order to enhance and complement the performance of the first algorithm did not lived up to expectations. However, despite of the algorithm efficiency and performance when compare with the first prototype, research later reveal its inadequacies as the scanning of database is done twice. Hence, there is an urgent need for the development of an improve algorithm that will address the inadequacies of the above discussed algorithms.

Considerations for the new algorithm

Having discussed the differences and inadequacies of the first and second algorithm such that in searching for frequent pattern pairs there would be need for the algorithm to construct Fp-tree before searching through the database while the second algorithm requires searching of database twice, which retrogressively make mining of database cumbersome. Based on these anomalies, the hybrid prototype targets is to ensure that:

1. the scanning of database is carried out once and not twice as in the case of first and second algorithm.
2. there will be no need for the construction of Fp-tree before scanning the database.
3. user's time in mining process is drastically reduced.

The New Algorithm-(Top-k FP-Support Gradient pruning)

Top-k FP-Support Gradient pruning is an improved technique for solving Multidimensional Constrained Gradient Mining problems in Transactional Database. (i.e finding frequent pattern pairs). Giving **p**romotion **c**onstraint $C_{pro,}$ we find a set promotion item (s) pÎP. Derive **g**radient **p**attern **t**hreshold C_{gpat} from C_{grad} and calculate Avgprofit(p) in transactional database. For each frequent pattern f found, form projected Multidimensional Database and then find Multidimensional patterns within the projected Databases in order to make scanning of database to be once. Compare the results to determine whether the frequent pattern f's Average(p) will passes C_{gpat} threshold and to report finally whether the result form a fre-

quent gradient fg or otherwise.

***Input*:** A multi-dimensional transaction database T, a significance constraint C_{sig},, a promotion constraint C_{pro}, a gradient constraint C_{grad}, and size of each bin

Output: The complete set of frequent gradient-promotion pattern pairs that satisfy the three constraints.

Method

(1) Given the database (T) for all item set.

i. If item set $\geq$ C_{sig} then

ii. Frequent patterns = item set

(2) Scan t once to get large frequent items using bin size.

(3) If (large frequent items > 0)
Derive gradient pattern threshold C_{gpat} from C_{grad} and AVGProfit (P) in T.

(4) FOR EACH frequent pattern f found, form projected MD – Database and then find MD – Patterns within projected databases.

a. If frequent pattern f's Average (P) passes C_{gpat} threshold

(i) Report as frequent gradient pattern f_g

ELSE

(ii) REPORT as No Frequent gradient pattern Nf_g

Implementation evaluation

The new system was implemented using C# because of the support for interactive applications.

Database Structure

The name and structures of Database are stated below:

SUPGRAP: It consists of two (2) main menus with additional command buttons on the toolbars, which serves as shortcut to all the menu items. Its General menu consists of authentic (real) data collected from supermarket stores are kept in this table. It has the following fields:

Transactions Item Transactions Entry Items Report Transaction Report Exit

2. TRANSACTION ITEM:-Transaction Items menu is a table displays where the user can enter the various item products being sold and their unique ID'S. it has the following fields:

Unique ID Item Name Date Time

3 TRANSACTION ENTRY FORM:- This is a table that allow the user gets to the screen where the various transactions can be entered on daily basis. It has the following fields:

Client Name Transaction Date Transaction Item Associated Profit

4. ITEMS REPORT:- This is a table that allow the user to get to the screen were the process criteria for generating report can be entered. It has the following field:

Period start period end Top-k Min. Length Prefix Item Constrained Gradient

5. TRANSACTION REPORT:- This is a table that shows the result of frequent gradient patterns generated with their equivalent Associated profit based on the selected process criteria. It has the following fields:

Transactions (Frequent Gradient Patterns) Associated Profit (N) Date Time

Experiments

As discussed earlier, the performance of TopkFpNoGp, TopkFp and *SUPGRAP* algorithms are affected by several factors such as the Significant Constraint, Gradient Con-

straint, Number of Tuples, transaction and pattern length, pattern length specified.
Hence, we take a closer look at how these factors actually affect the performance of the three algorithms to examine which, and under what situation, one is better than the other, taking into consideration the run-time speed and amount of memory used.

Evaluation Results

Results on Top-k FP-growth with Support, Gradient and Support Gradient Pruning

For multi-dimensional constrained gradient mining in databases, we tested Top-k FP-Growth with No Gradient pruning (TopkFPNoGp), Top-k FP-growth with Gradient pruning (TopkFP) and Top-k FP-growth with Support Gradient Pruning (SupGrap). We conducted experiments on synthetic datasets generated from a synthetic transaction database.

In Figure 3, we tested the effect of significance constraint Csig on runtime. We fix gradient threshold Cgrad to 1.1, average transaction length to 12, average pattern length to 6. As we can see, when Csig decreases, Top-k FP-growth with No Gradient Pruning (TopkFPNoGP) requires significantly more time than Top-k Fp-growth with Gradient Pruning (TopkFP). And Top -k Fp-growth with Gradient Pruning (TopkFP) requires more time than Top-k Fp-growth with Support Gradient Pruning (SUPGRAP) which shows that Support Gradient Pruning is an improved method reason been that it out performs the other two methods. This is because without gradient constraint pruning, the superset of frequent patterns quickly outgrows the subset of frequent gradient patterns. The time required to mine the Top-k FP-tree increases dramatically and the post-processing time required to filter the results also increases.

Next we investigated the effect of gradient constraint Cgrad on algorithms' runtime. We fix Csig to 25 (0.25% of total number of tuples) but range Cgrad from 1 to 10. From Figure 4, we conclude that Top-k FP-growth with Support Pruning requires pretty much a constant amount of time because the number of frequent patterns is constant as Csig is fixed. However, Top-k FP-growth with Support Gradient Pruning is better in performance when compared with other two algorithms.

As shown in figure 5, we fix Csig, Cgrad and number of beans as we change the number of tuples. Intuitively, as the numbers of tuples increases, the runtime for three algorithms also increases. However, Top-kFP-growth with No Gradient pruning takes more time than Top-kFP-growth whileTop-kFP-growth with support Gradient pruning takes lesser compare to other two algorithms as the number of tuples increases. This is to the increase in the number of frequent patterns satisfying Csig and thus more time in mining those frequent patterns that may satisfy Cgrad.

For transaction databases, we also performed an experiment to determine the effect of average transaction length and average pattern length. First, we test the effect of average transaction length by generating multiple datasets of size 10,000 tuples with different average transaction lengths. Note that we fix the average pattern length to be half of average transaction length such that the frequent patterns length we mine increases with average transaction length. In Figure 6, we can see as the average transaction (and thus, pattern) length increases, both algorithms re-

quire more time to mine longer frequent gradient patterns as expected. The gradient pruning strategy in Top-k FP-growth Support Gradient Pruning algorithm performs better than Top-k FP-growth with Gradient Pruning and this also performs better than Top-k FP-growth with Support Pruning since more search space is pruned.

In Figure 7, we then showed the effect of average pattern length on runtime of both algorithms. As expected, when average pattern length increases, the amount of time to mine longer frequent patterns increases. The runtime for Top-k FP-growth with No Gradient Pruning increases significantly since other two uses gradient constraint for pruning to decrease the number of frequent patterns generated but Top-k FP-growth with support Gradient Pruning use lesser time compare to others.

FUTURE WORK

This work confirms our recognition of the potential in constrained gradient analysis. We see that further research effort is worth investing into this problem area in order to solve more related problems. Some of the suggested future works are:

1. Instead of returning the complete answer set, the algorithm can be modified to generate and rank the best-N answers that maximize the difference in measures. This helps users to easily understand the result.
2. Based on current definition of the problem, users must identify the items of interest for promotion, i.e. the promotion items. This suits the scenario when a user is looking to promote particular items based on festivities, inventory status and item popularity. Instead the algorithm can be modified to suggest an item category for promotion and identify the cross-selling items to achieve the most profit.
3. Investigating into the possibility of using CLOSET algorithm (Jiawei et al., 2000) to mine a closet set of pattern pairs.

CONCLUSION

In this work, an improved algorithm and data structure were proposed to solve the problem in transaction database environment. The original FP-tree was modified to become a Top-k FP-tree data structure for storing average measure information of transactions. FP-growth algorithm was enhanced to suit the Top-k FP-tree data structure. Integrating gradient constraint into processing, Top-k FP-growth with Support Gradient algorithm is more effective in its pruning strategy than Top-k FP-growth with Gradient Pruning and Top-k FP-growth with Support Pruning algorithm.

This work has been used to solve problem on multi-dimensional constrained gradient mining in transactional database. Combining FP-growth and H-cubing algorithms and their corresponding data structures, FP-tree and H-tree respectively, we propose a Top-k FP-growth algorithm on a top-k FP-tree hybrid data structure for mining multi-dimensional constrained gradients in transaction databases.

With increasing competition in the business world, leveraging company data to give a competitive advantage is a sensible and important step of survival. The direct application of this problem in the business paradigm warrants the significance of this research effort. Experimental evaluations in this paper bring promises to this area. Our proposed algorithms successfully and efficiently extract multi-dimensional entity pairs that have significance difference in their measures.

Table 2: Result of frequent gradient patterns of figure 1

Mining with prefix path as…	Resulting frequent gradient patterns...
(p)	(p: 3, 113.3), (cp: 3, 113.3)
(m)	(m: 3, 120), (fm: 3, 120), (cfm:3, 120), (afm:3,120), (acfm:3, 120), (cm:3, 120), (acm:3, 120), (am:3, 120)
(f)	(cf:3, 120), (acf:3, 120), (af:3, 120)
(c)	(c:4, 110, (ac:3, 120)
(b)	NONE
(a)	(a:3, 120)

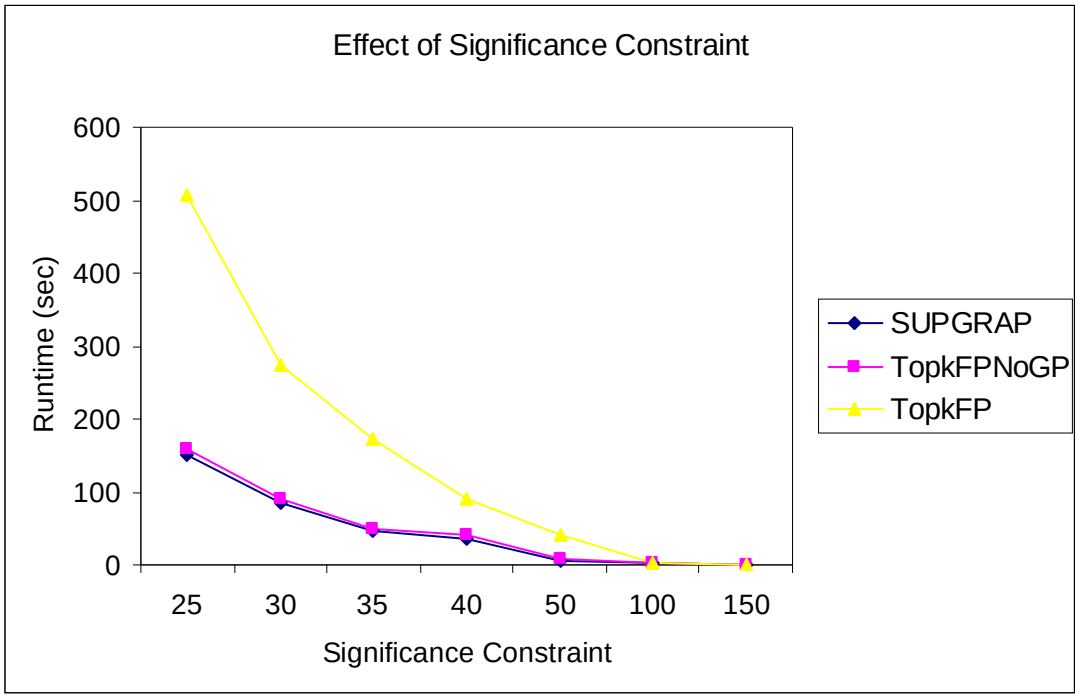

Figure 3: Effect of Significance Constraint

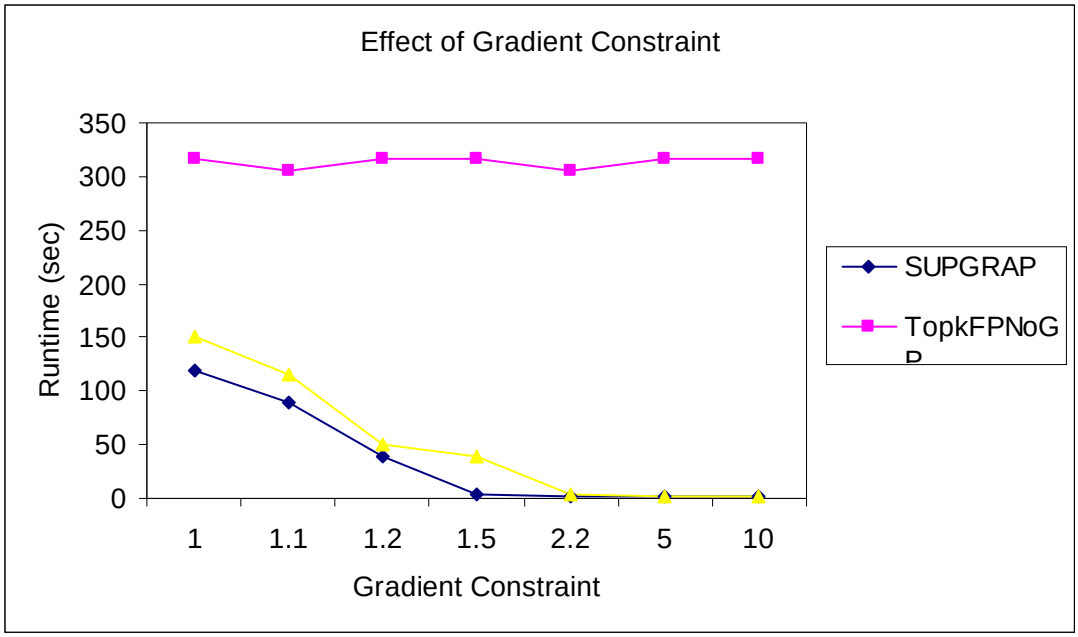

Figure 4: Effect of Gradient Constraint

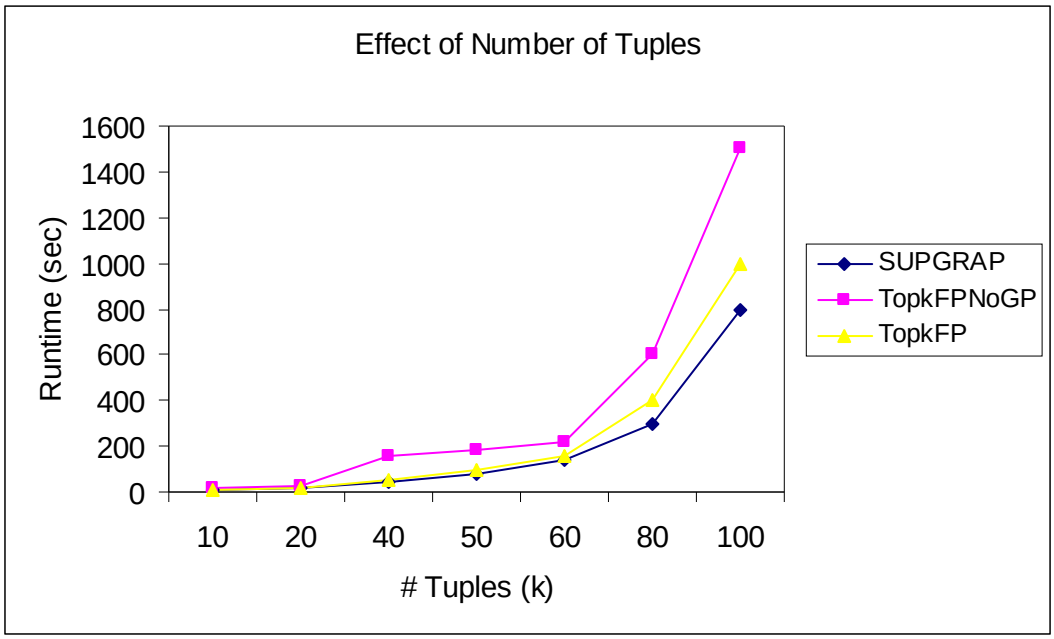

Figure 5: Effect of Number of Tuples

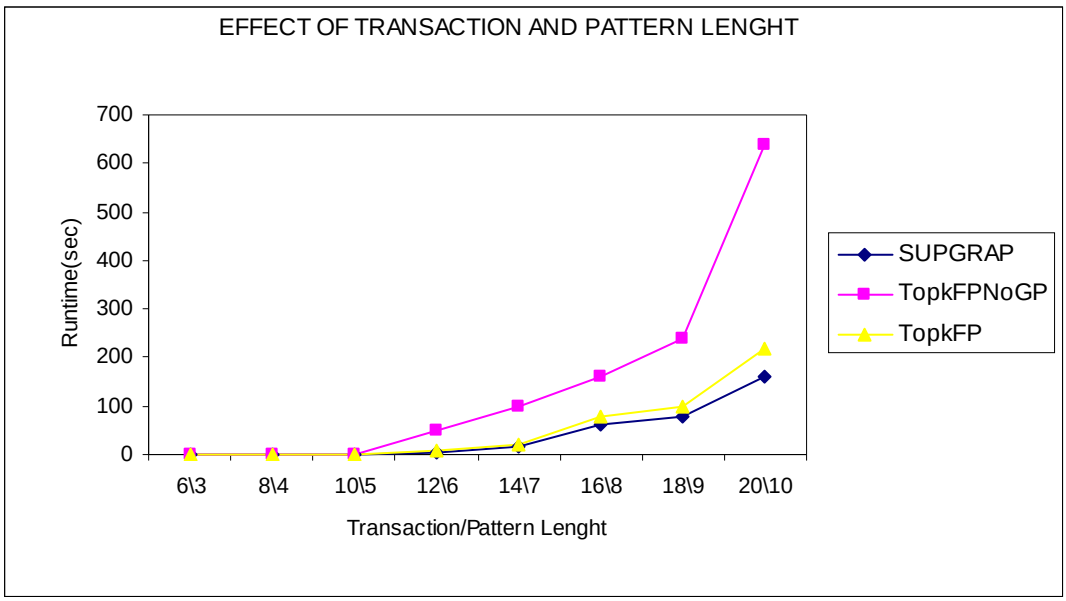

Figure 6: Effect of Transaction and Pattern Lengths

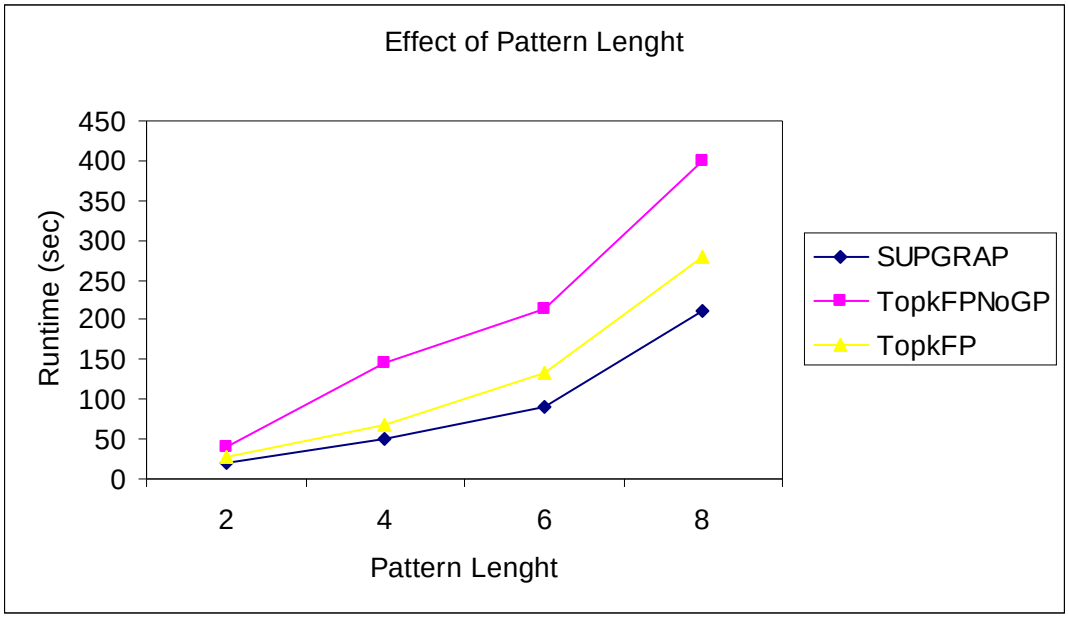

Figure 7: Effect of Pattern Length

REFERENCES

Agrawal, R., Ramakrishnan, S. 1994. Fast Algorithms for Mining Association Rules. In Proceedings of the *20th International Conference on Very Large Data bases (VLDB'04),* Santiago, de Chile, Chile.

Agrawal, R., Imielinski, T., Swami, A. 1993. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'93), P*. 207-216, Washington, D.C.

Guozhu Dong, Jiawei Han, Joyce M.W. Lam, Jian Pei, Ke Wang 2001. Mining Multidimensional Constrained Gradients in Data Cubes. To appear in Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01), September 11-14, 2001, Roma, Italy.

Guozhu Dong, Jiawei Han, Joyce M.W. Lam, Jian Pei, Ke Wang, Wei Zou 2004. "Mining Constrained Gradients in Large Databases," *IEEE Transactions on Knowledge and Data Engineering*, 16(8): 922-938.

Ibrahim, S.A. 2004. An Efficient pattern Growth Mining of Closed Frequent Itemsets: A thesis Submitted in Fulfilment of the requirement for the Degree of Master of science in the Department of Mathematical sciences, University of Agriculture, Abeokuta.

Jian Pei, Jiawei Han 2001. Can we push more Constraints into Frequent Pattern Mining?.In *Proceedings of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00),* p.350-354, August 20-23, 2000, Boston, MA, USA.

Jian Pei, Jiawei Han, Laks V.S. Lakshmanan 2001. Mining Frequent Itemsets with Convertible Constraints. In: *Proceedings of the 17th International Conference on Data Engineering (ICDE'01),* April 2-6, 2001, Heidelberg, Germany.

Jiawei Han, Jian Pei, Yiwen Yin 2000. Mining Frequent Patterns without Candidate Generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00),* May 16-18, 2000, Dallas, Texas, USA.

Joyce Man Wing Lam 2001: Multidimensional Constrained Gradient Mining. A Thesis Submitted in Partial Fulfillment the Requirement for the Degree Master of Science in the School of Computing Science: Simon Fraser University.

Neinbe, O.J. 2004. Mining Frequent MAX-Sequential patterns from Customer Portifolio – Database: A thesis submitted in Fulfilment of the requirement for the Degree of Master of science in the Department of Mathematical sciences, University of Agriculture, Abeokuta.

Ansari, S., Sadreddini, M.H. 2009. An Efficient Approach to Mining Frequent Itemsets on Data Streams. In: *Proceedings of World Academy of Science, Engineering and Technology, 37:* ISBN 2070-3.

Tomasz Imielinski, L. Khachiyan, A. Abdulghani 2002. "Cubegrades: Generalizing Association Rules," *Data Mining and Knowledge Discovery,* 6: 219-258.

Wang, J., Han, J., Pei, J. 2006. Closed Constrained Gradient Mining in Retail Databases, *IEET Transactions on Knowledge and Data Engineering,* 18(6): 764-769.

(Manuscript received: 16th February, 2009; accepted: 23rd February, 2010).