MINING MOST FREQEUENT CLOSED ITEMSETS USING BEST FIRST SEARCH

PANIDA SONGRAM

A THESIS SUBMITTED IN FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2009

KMITL-2009-SC-D-002-047

หัวข้อวิทยานิพนธ์	การสืบค้นกลุ่มข้อมูลแบบปิดที่เกิดบ่อยที่สุดโดยใช้การค้นหาแบบดีที่สุดก่อน
นักศึกษา	นางสาวพนิดา ทรงรัมย์
รหัสประจำตัว	49062902
ปริญญา	ปรัชญาดุษฎีบัณฑิต
สาขาวิชา	วิทยาการคอมพิวเตอร์
พ.ศ.	2552
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.วีระ บุญจริง

บทคัดย่อ

งานวิจัยนี้ได้ศึกษาทฤษฎีและขั้นตอนวิธีการสืบค้นกลุ่มข้อมูลแบบปิดที่เกิดบ่อยที่สุด ซึ่งการสืบค้นกลุ่มข้อมูลแบบปิด k ลำดับแรกและการสืบค้นกลุ่มข้อมูลแบบปิดที่น่าสนใจ N ลำดับแรกเป็นงานวิจัยที่ถูกนำเสนอขึ้นเพื่อสืบค้นจำนวนกลุ่มข้อมูลแบบปิดที่เกิดบ่อยที่สุดโดยไม่ต้องกำหนดค่าสนับสนุนขั้นต่ำ งานวิจัยดังกล่าวนอกจากจะหลีกเลี่ยงการสร้างกลุ่มข้อมูลที่ซ้ำซ้อนแล้วยังอำนวยความสะดวกให้กับผู้ใช้ที่ต้องการสืบค้นจำนวนกลุ่มข้อมูลแบบปิดที่เกิดบ่อยที่สุด งานวิจัยนี้ได้นำเสนอสองขั้นตอนวิธีการที่มีประสิทธิภาพสำหรับสืบค้นกลุ่มข้อมูลแบบปิด k ลำดับแรกและกลุ่มข้อมูลแบบปิดที่น่าสนใจ N ลำดับแรกโดยใช้การค้นหาที่ดีที่สุดก่อน ซึ่งทำให้สามารถสร้างกลุ่มข้อมูลแบบปิดตามค่าสนับสนุนจากมากไปน้อย ดังนั้นจึงไม่จำเป็นต้องหาค่าสนับสนุนตัวสุดท้ายก่อนกระบวนการสืบค้น ซึ่งนำไปสู่การตัดกลุ่มข้อมูลที่ไม่จำเป็นออกและหยุดกระบวนการสืบค้นได้อย่างรวดเร็ว นอกจากนี้กลุ่มข้อมูลแบบปิดยังถูกสร้างโดยไม่ต้องเก็บกลุ่มข้อมูลคู่แข่งในหน่วยความจำโดยการคำนวนหาโคลสเซอร์ของกลุ่มข้อมูล ยิ่งไปกว่านั้น ขั้นตอนวิธีการที่นำเสนอยังหลีกเลี่ยงการคำนวณที่ซ้ำซ้อนซึ่งนำไปสู่กลุ่มข้อมูลแบบปิดตัวเดียวกันโดยการตรวจสอบและขจัดกลุ่มข้อมูลที่ซ้ำซ้อนออก

Thesis Title	Mining Most Frequent Closed Itemsets Using Best First Search
Student	Miss Panida Songram
Student ID.	49062902
Degree	Doctor of Philosophy
Programme	Computer Science
Year	2009
Thesis Advisor	Assoc. Prof. Dr. Veera Boonjing

ABSTRACT

This thesis studies theories and algorithms for mining most frequent closed itemsets. Top-k closed itemset mining and N-most interesting closed itemset mining are interesting topics proposed for mining the desired number of the most frequent closed itemsets without giving a minimum support threshold value. They not only avoid the redundant itemsets generated but also give convenience for users to retrieve the desired number of closed itemsets. Two efficient algorithms are proposed in this thesis for mining top-k closed itemsets and N-most interesting closed itemsets by using best first search. The proposed algorithms mine closed itemsets in descending order of their supports, so finding a final threshold before actual mining phase are unnecessary. This approach also leads to an efficient pruning of unnecessary itemsets and stops mining rapidly. Moreover, the proposed algorithms generate closed itemsets without keeping candidates in memory by computing closure of itemsets. Furthermore, they avoid redundant computation of the same closed itemset by detecting and discarding duplicated itemsets.

ACKNOWLEDGEMENTS

The author would like to thank my family, especially my parents, for giving opportunities, pushing me, understanding me, and helping me to achieve my Ph.D.

I would like to express my deeply many thanks to Assoc. Prof. Dr. Veera Boonjing for all advises and very good support me concerning about my papers and thesis.

I would like to thank Office of Academic Administration of King Mongkut's Institute of Technology Ladkrabang, Mahasarakham University, Higher Education Commission, and National Centre of Excellence in Mathematics for supporting my Ph.D. study.

I would like to thank my friends and Dr. Richard Booth for useful comments and proof-reading the English.

Lastly I would like to thank my colleagues and coworkers in Mahasarakham University for supporting my Ph.D. study.

Panida Songram

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT)

LIST OF TABLES

LIST OF FIGURES

Figures Page

CHAPTER 1

Introduction

1.1 The Problem of Most Frequent Closed Itemset Mining

Frequent pattern mining has been extensively studied in data mining. This mining determines patterns from a database through a minimum support threshold. Frequent itemset mining is a popular problem in frequent pattern mining. It is exploited in many applications that are interested in how often two or more objects of interest co-occur such as the analysis of customer behavior, and web access. Since frequent itemset mining has to find itemsets whose supports are not less than a minimum support threshold, it may give a large number of redundant itemsets if the threshold is very low or a large database is mined. To solve this problem, closed itemset mining was proposed to reduce the large number of itemsets generated without information loss. The idea of closed itemset mining is that considers only frequent itemsets having no superset with the same support. This idea was also adapted for mining sequences, called closed sequence mining. Similar closed itemset mining, closed sequence mining was proposed to reduce a large number of redundant sequences generated in frequent sequence mining. However, closed itemset mining and closed sequence mining have to face the difficulty of specification of an appropriate minimum support threshold. Finding an appropriate threshold is very hard for users having no knowledge of mining query and task-specific data. A large of resulting itemsets may be generated if users give a too small threshold, whereas no answer is produced if a too large threshold is given. Top-k itemset mining and N-most interesting itemset mining were proposed as alternative mining tasks to avoid the difficulty of specification of an appropriate threshold. They allow users to retrieve a desired number of the most frequent itemsets which is the number of itemsets having highest supports. The number of the most frequent itemsets is generated without providing the threshold value. Although avoiding the difficulty of specification of an appropriate threshold, both mining tasks still give redundant itemsets. Therefore, the idea of closed are included with top-k itemset mining, called top-k closed itemset mining, to avoid redundant itemsets generated. Top-k closed itemset mining was extended for mining top-k closed sequences. Many efficient algorithms were proposed in top-k itemset mining, N-most interesting itemset mining, top-k closed itemset mining and top-k closed sequence mining.

To retrieve the desired number of most frequent itemsets without giving the threshold value, the most algorithms have to find the final threshold value before actual mining phase. In addition, they need to maintain candidates to ensure that they are real answer.

1.2 Research Objective

This thesis is proposed to study theories and algorithms for mining most frequent closed itemsets. We propose a new efficient algorithm, called TOPK_CLOSED, to mine top-*k* closed itemsets. The algorithm discovers most frequent closed itemsets by employing a best first search strategy. Moreover, it generates closed itemsets without keeping candidates by using closure operator. In this thesis, we also propose an alternative task of mining, called *N*-most interesting closed itemset mining. *N*-most interesting closed itemset mining allows users to retrieve a desired number of non-redundant itemsets in different lengths without providing a minimum support threshold of each length. Moreover, input parameters in this task are convenient for users to control with their applications. The efficient algorithm, called NCLOSED, is proposed for mining *N*-most interesting closed itemsets by adapting the TOPK_CLOSED algorithm.

1.3 Scope of Thesis

This thesis studies theories and algorithms for mining most frequent closed itemsets. Two efficient algorithms are proposed in this thesis for mining top-*k* closed itemsets and *N*-most interesting closed itemsets. In top-*k* closed itemset mining, we investigate experiments by comparing running time of TOPK_CLOSED with the TopKMiner algorithm on dense and sparse data sets. In *N*-most interesting closed itemset mining, experiments are investigated to consider the performance of NCLOSED on dense and sparse data sets with different sizes and different input parameters. The data sets used in our experiments consist of real data sets and a syntactic data set. The real data sets (mushroom, connect, chess, pumsb and gazelle) are download at http://fimi.cs.helsinki.fi/. A synthetic data set (T10I4D8K) is generated from IBM generator.

1.4 Organization of Thesis

The remainder of this thesis is organized as follows. In Chapter 2, we mention tasks of mining related to most frequent closed patterns. Chapter 3 presents two efficient algorithms for

mining top-k closed itemsets and N-most interesting closed itemsets. In Chapter 4, it shows experimental evaluation. Finally, conclusion and recommendation are given in Chapter 5.

CHAPTER 2

Literature Review

This chapter addresses mining tasks related to the most frequent closed patterns. We group the mining tasks into two categories. The first category concerns with itemset mining as presented in section 2.1. The second one concerns with sequence mining as given in section 2.2.

2.1 Itemset Mining

Frequent itemset mining is the first sub-problem of association rule mining. It was introduced by R. Agrawal et al. [9]. All frequent itemsets are used to produce associate rules. A large number of redundant itemsets may be generated. As a consequence, a large of association rules is produced and then makes the task of analysts very complex. To reduce the problem, closed itemset mining was introduced by N. Pasquier et al. [26, 27, 28] to eliminate redundant itemsets generated in frequent itemset mining. It finds closed itemsets instead of frequent itemsets by using a closure mechanism based on Galois connection [26, 27, 28] as explained in section 2.1.1. Many efficient algorithms were proposed for mining closed itemsets such as CLOSE, CLOSET, CLOSET+, FPCLOSE+, CHARM, DCI-CLOSED and LCM. The algorithms are briefly explained in section 2.1.2. Moreover, alternative tasks were proposed to retrieve most frequent patterns without giving the support threshold value in mining process such as top-k itemset mining, top-k closed itemset mining, top-k closed sequence mining, and N-most interesting itemset mining. Efficient algorithms of them are reviewed in section 2.1.3 and 2.1.4.

2.1.1 Galois Connection

Definition 2.1 [26, 27, 28] A data mining context is a triple $C=(D,I,R)$. D is a non-empty finite set of all transaction ids in the database, and I is a non-empty finite set of distinct items appearing in the database. $R \subseteq D \times I$ is a binary relation between transaction ids and items. Each couple $(d,i) \in R$ denotes the fact that the transaction id $d \in D$ has the item $i \in I$.

Definition 2.2 [26, 27, 28] Let $C=(D,I,R)$ be a data mining context, a tidset (a set of transaction ids) T be a non-empty subset of D and an itemset (a set of items) X is a non-empty subset of I. The concept of closed itemset mining based on two functions f and g [4, 5]:

$$f(T)=\{i\in I\mid \forall d\in T,(d,i)\in R\} \tag{2.1}$$

$$g(X)=\{d\in D\mid \forall i\in X,(d,i)\in R\} \tag{2.2}$$

The function f returns the largest itemset included in all the tidset belonging to T, and the function g returns a tidset supporting a given itemset X.

Definition 2.3 [4, 5, 26, 27, 28] An itemset X is closed if and only if the closure of X $\varsigma(X)=f(g(X))=fog(X)=X$, where the composite function $\varsigma=fog$ is called a Galois operator or closure operator.

Given Galois connection (f, g) and $\varsigma'=gof$, the following properties hold for all X_1, X_2 and T_1, T_2 [26, 27, 28].

$$X_1\subseteq X_2\rightarrow g(X_1)\supseteq g(X_2) \tag{2.3}$$

$$X_1\subseteq \varsigma(X_1) \tag{2.4}$$

$$\varsigma(\varsigma(X_1))=\varsigma(X_1) \tag{2.5}$$

$$X_1\subseteq X_2\rightarrow \varsigma(X_1)\subseteq \varsigma(X_2) \tag{2.6}$$

$$\varsigma'(g(X))=g(X) \tag{2.7}$$

$$T_1\subseteq T_2\rightarrow f(T_1)\supseteq f(T_2) \tag{2.8}$$

$$T\subseteq \varsigma'(T) \tag{2.9}$$

$$\varsigma'(\varsigma'(T_1))=\varsigma'(T_1) \tag{2.10}$$

$$T_1\subseteq T_2\rightarrow \varsigma'(T_1)\subseteq \varsigma'(T_2) \tag{2.11}$$

$$\varsigma(f(T_1))=f(T_1) \tag{2.12}$$

$$T_1\subseteq g(X_1)\leftrightarrow X_1\subseteq f(T_1) \tag{2.13}$$

Example 2.1 Consider table 2.1 [5], the letters *a-d* represent items in the database.

Table 2.1 An itemset database

TID	Items
1.	*b d*
2.	*a b c d*
3.	*a c d*
4.	*c*

In this thesis, we represent an itemset in form $i_1 i_2 ... i_k$ instead of $\{i_1, i_2, ..., i_k\}$. Considering item *a*, it is not closed because $\varsigma(a) = f(g(a)) = f(\{2, 3\}) = acd$. Considering itemset *acd*, it is closed because $\varsigma(acd) = f(g(acd)) = f(\{2, 3\}) = acd$.

In figure 2.1 [5], closure operator is used to determine closed itemsets. A closed itemset can be found by calculating a closure of an itemset. The closure operator defines a set of equivalence classes over the lattice of itemsets: two itemsets belong to the same equivalence class if and only if they have the same closure. For example, *a* and *acd* are in the same equivalence class, because $\varsigma(a) = \varsigma(acd)$. Therefore, the itemsets with the same closure are grouped in the same equivalence class. Each equivalence class contains elements sharing the same support, and closed itemsets are maximal elements of each equivalence class. We can see that closed itemsets can cover all elements in the same class.

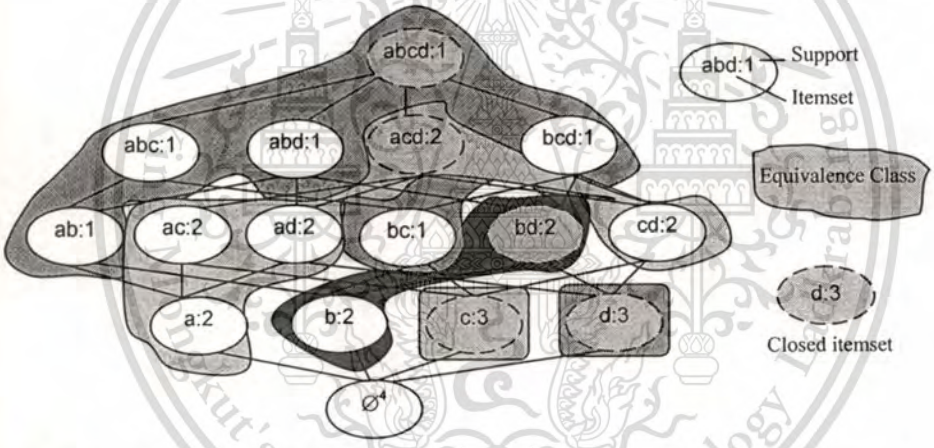


Figure 2.1 Lattice of all the itemsets with closed itemsets and equivalence classes

2.1.2 Closed Itemset Mining

Many efficient algorithms were proposed for mining closed itemsets. The CLOSE or A-CLOSE algorithm is the first algorithm for mining closed itemsets. This algorithm generates closed itemsets by finding closure of smallest frequent itemsets of all equivalence classes, called generators. CLOSE first identifies generators. If an itemset whose support is the same as support of any its subsets, such itemset will be pruned because it cannot be a generator. Then, the closure of each generator is computed by intersection of all transactions containing such generator. This way may lead to redundant computation because two generators may lead to the same closed itemsets. For example, closed itemset *abcd* has to be computed twice because it is closure of two

generators, *ab* and *bc* (see figure 2.1). The closure operation is also expensive if there are many transactions containing a generator. Moreover, it needs to maintain the already mined frequent itemset in order to identify generator.

To avoid redundant computation of the same closed itemsets. The CLOSET and CLOSET+ algorithms adopt the FP-tree structure (Frequent Pattern tree structure) in the FP-GROWTH [12] algorithm to mine closed itemsets. An FP-tree is constructed to compress all the transactions in a database which is stored in main memory. Infrequent items are not used to construct the tree and a set of transactions sharing the same subset of items may share common prefix paths from the root in the FP-tree. Both algorithms use divide-and-conquer technique to mine closed itemsets. They split the context in an FP-tree into smaller sub-context and recursively perform the closed itemset process on those sub-contexts. CLOSET mines closed itemsets by closure climbing and growing up closed itemsets with items having the same support. It verifies closed itemset by using subset checking, if a newly found itemset is a subset of an already found closed itemset candidate with the same support, the newly found is not a closed itemset. CLOSET was improved to CLOSET+. CLOSET+ uses different methods for mining different data sets. It mines closed itemsets from an FP-tree in a top-down manner for sparse data sets and bottom-up manner for dense data sets. During the mining process, CLOSET+ uses the item merging, item skipping, and sub-itemset pruning methods to prune search space. For verifying closed itemsets, it uses upward checking method for sparse data sets and adopts subset checking in CLOSET for dense data sets. FPCLOSE is another algorithm exploiting an FP-tree to mine closed itemsets. It stores the previously mined closed itemsets in recursively constructed a CFI-tree (Closed Frequent Itemset tree). Consequently, subset checking cost is less expensive than that of CLOSET and CLOSET+. In addition, a simple array structure is used to reduce the traversal time.

Unlike CLOSET, CLOSET+ and FPCLOSE, the CHARM algorithm explores the search space in depth-first manner technique without splitting the context in FP-tree into smaller sub-contexts. It proposed a new technique to mine closed itemsets by exploring vertical data format. CHARM not only exploits itemset space but also transaction space to find closed itemsets. CHARM introduces a data structure, called IT-tree (Itemset Tidset tree). Each node in the IT-tree contains a frequent itemset and tidset to which it belong. As soon as a frequent itemset is generated, its tidset is compared with those of the other itemsets having the same parent. If one tidset includes another one, the associated nodes are merged. Since tidset may be large, intersection computation becomes large. To fast the computation, CHARM stores diffsets in each

node instead of tidset, except for the root class which use tidset. In addition, the number of mismatches in both the diffesets is used for subset checking.

The previous algorithms, CLOSET, CLOSET+, FPCLOSE and CHARM, enumerate closed itemsets by pruning unnecessary itemsets. They need to keep the already mined closed itemset candidates in order to do subset checking. If a large number of candidates is generated, they consume much memory to remove non-closed itemsets. Moreover, some of the algorithms, CLOSE, A-CLOSE and CHARM algorithm, lead to compute the same closed itemset. To avoid these problems, the DCI-CLOSED and LCM algorithms were proposed to generate closed itemsets without storing candidates. In addition, they can avoid the redundant computation of the same closed itemset.

DCI-CLOSED was improved from CHARM. It adopts tidset and the hybrid traverse of the search space in CHARM. DCI-CLOSED uses closure climbing to brow the search space, find generators and compute their closure. This method allows discarding generators whose closures were already mined. It can detect and discard duplicate generators based on lexicographic order relation. It also finds closed itemsets without keeping the set of mined closed itemsets in main memory. After discovering a closed itemset f, a new itemset g is grown by extending f with a frequent item i, $i \notin f$. If the tidset of g is subset of those of any preceding frequent items, g is not generator. Otherwise, g is an order preserving generator of a new closed itemset. Then closure of the generator is computed to find a closed itemset by unifying it with the post frequent items whose tidset includes tidset of the generator. LCM (Linear time Closed pattern Miner) is similar to DCI-CLOSED. It finds closed itemsets from transaction database by calculating closures of generators without storing already mined closed itemsets. Nevertheless, LCM calculates a closure in the different strategies. LCM proposed prefix-preserving closure extension of closed itemsets to search all frequent closed itemsets, a new closed itemset is extended from previously obtained closed itemset.

2.1.3 Top-k Itemset Mining and Top-k Closed Itemset Mining

Top-k itemset mining was proposed to retrieve a desired number of the first k most interesting frequent itemsets without providing the threshold value. Y. Hirate et al. proposed the TF^{2}P-growth algorithm [42] by extending the FP-GROWTH algorithm. This algorithm also uses an array to trace the potential patterns for the final top-k itemsets. However, TF^{2}P-growth is an exhaustive approach, which causes a slowdown in performance [36]. The ExMiner algorithm [33]

was proposed by T.M.,Quang et al. to solve the problem of TF^2P-growth by finding a final minimum support threshold (or the appropriate threshold that can retrieve the desired number of resulting patterns) before the actual mining phase. It constructs an FP-tree to raise the threshold and uses it to prune the tree before mining. Unfortunately, it requires a large number of items to find the final threshold. Thus, T.M.,Quang et al. proposed an optimized algorithm, OExMiner [37], to reduce the large number of items. Chung et al. proposed the MTK (standing for the Memory-constraint top-k frequent-pattern mining) algorithm [19] by using the efficient search approach, $\delta - stair$, to limit the number of candidates. Moreover, it integrates many techniques to efficiently mine top-k itemsets such as the scan-reduction technique and hash-indexing technique. However, MTK has to store candidates to check they are real answers.

Although top-k itemset mining can avoid the difficulty of specification of an appropriate minimum support threshold, it still gives redundant itemsets. Wang et al.[13, 16] proposed top-k closed itemset mining to avoid redundant itemsets generated. They proposed to mine top-k closed itemsets with minimal length. The TFP algorithm was developed to mine such patterns. It first constructs an FP-tree to raise the threshold value before mining phase. Therefore, it has to face the problem like ExMiner. It is very inefficient in memory consumption [3]. Moreover, TFP has to prune unnecessary itemsets from the tree after the final threshold is found. It is clear that it wastes the time before actual mining. In addition, TFP has to maintain candidates to ensure that they are real closed. The MTK_Closed algorithm [19] was extended from MTK to mine top-k closed itemsets. Like MTK, MTK_Closed has to store candidates to check they are real closed. Pietracaprina et al. proposed the TopKMiner algorithm [1] for mining top-k closed itemsets. TopKMiner combines the LCM algorithm [38, 38] and priority queue to mine top-k closed itemsets. LCM is exploited to generate closed itemsets without keeping candidates. Priority queue is used to keeps entries corresponding to closed itemsets. Closed itemsets in LCM are generated in order of decreasing support. This idea is supported by C. Wu [6]. He proved that a heuristic algorithm is preferred to solve top-k closed itemset mining over an exact algorithm. Unlike TFP and MTK_CLOSED, TopKMiner allows users to dynamically raise the value of k without restarting computation.

2.1.4 *N*-most Interesting Itemset Mining

An interesting mining task, called *N*-most interesting itemset mining, was also introduced by C. Fu et al. to avoid the difficulty of specification of a minimum support threshold. *N*-most

interesting itemset mining generates N l-itemsets with the highest supports for 1 up to a certain l_{max} value, where l_{max} the upper bound of the length of itemsets, and N is the desired number of l-itemsets. The Itemset-Loop algorithm [2] was developed to mine N-most interesting itemsets. This algorithm is based on the Apriori algorithm [31]. It uses a candidate generation-and-test mechanism.

Cheung et al. [43, 44] proposed three algorithms, LOOPBACK, BOLB, and BOMO, for mining N-most interesting itemsets by adapting FP-tree structure. BOMO builds a complete FP-tree with all items in a database to find a final threshold of each length. LOOPBACK builds an FP-tree and initializes a final threshold to be the support of the Nth sorted largest 1-frequent. If a number of any l-itemset is less than N, the tree will be rebuilt to find the smaller threshold in order to mine more itemsets in the mining phase. BOLB is a hybrid approach of BOMO and LOOPBACK. Like BOMO, it builds an complete FP-tree only once. The mining process is applied from the technique of LOOPBACK.

Arshad et al.[25] proposed two algorithms, NFOLD-growth and LOOPBACK-NFOLD-growth, to mine N-most interesting itemsets. Both algorithms are based on a SOTrieIT structure (Support-Ordered Trie Itemset structure) to quickly find N-most interesting 1-itemsets and 2-itemsets. Both N-most interesting itemsets are used to prune transactions for creating a complete FP-tree. Then N-most interesting l-itemsets, l>2, are mined. The mining process of NFOLD-growth is similar to BOMO, while the mining process of LOOPBACK-NFOLD-growth likes that of LOOPBACK.

S. Ngan et al. [35] proposed the COFI-BOMO algorithm to mine N-most interesting itemsets. COFI-BOMO is similar to BOMO, but it is improved by using a COFI-tree structure (Co-occurrence Frequent Item Tree structure) [21]. The COFI-tree structure is adopted to build a number of independent and relatively small trees for each item.

2.2 Sequence Mining

Frequent sequence mining is an important data mining task. It is used in many applications such as analysis of customer purchase, web-access, discovery of motifs and tandem repeats in DNA sequences, analysis of various sequencing or time-related process, scientific experiments, disease treatments, natural disasters [31, 32]. Frequent sequence mining is similar to frequent itemset mining, but the order of itemsets in each sequence is important. Therefore,

frequent sequence mining requires ordered matching for subsequence testing with is more difficult than simple subset testing in frequent itemset mining. Since frequent sequence mining generate a large number of redundant sequences, closed sequence mining was proposed to avoid this problem by adapting the idea of closed in closed itemset mining. Efficient algorithms for mining closed sequences are briefly mentioned in section 2.2.1. Moreover, we mention multidimensional sequence mining which was proposed in different propositions in section 2.2.2.

2.2.1 Closed Sequence Mining

The CLOSPAN algorithm [41] is the first algorithm introduced to mine closed sequences. CLOSPAN first generates a set of closed sequence candidates and stores them in a hash indexed result-tree structure, and then non-closed sequences are eliminated. This algorithm proposed backward sub-pattern and backward super-pattern methods to prune search space. To avoid database scans, some sequences are kept to be included in some other ones. Moreover, CLOSPAN has to store candidates for checking closed sequences during the mining process. Unfortunately, if a minimum support threshold is low or sequences in database become long, the large number of closed sequence candidates occupies much memory and lead to large search space for the subset checking of a new closed sequence.

The BIDE algorithm [15] was proposed to mine sequence data set consisting of events containing a single item. It can generate closed sequences without keeping a set of candidates in memory. BIDE builds a projected datasets and find frequent sequences. Then, the frequent sequences are checked whether they are closed by using bi-directional extension. The forward extension event checking in bi-directional is used to grow a new sequence. BIDE is further improved by using back scan pruning method and scan skip optimization techniques to prunes the search space. However, the closed sequence generation and the search space pruning need to check sequences in all the transactions. If a database consists of a lot of transactions, it may take an expensive cost to do the closed sequence generation and search space pruning.

2.2.2 Multidimensional Sequence Mining and Closed Multidimensional Sequence Mining

Multidimensional sequence mining is an important topic in data mining. Many researchers have studied this topic but different propositions. The first proposition of multidimensional sequence mining was proposed by C. Yu et at.[7]. This proposition mines

sequence from multidimensional sequence data containing several dimensions. The AprioriMD and PrefixMDSpan algorithms were proposed in this proposition by applying the Apriori and PrefixSpan algorithms, respectively.

The second proposition was proposed by M. Plantevit et al. [24]. A Multidimensional sequence in this proposition is sequences containing several dimensions combined over time. The M2SP algorithm was proposed to mine such sequences. Moreover, the MD-CLOSED algorithm [23] was developed to reduce the amount of discovered patterns in M2SP by mining only closed multidimensional sequences. Like CLOSPAN, MD-CLOSED firstly generates candidates and prunes non-closed later. MD-CLOSED applies lexicographic order to LGS order (LexicoGraphic Specialization order) for mining closed multidimensional sequences in depth-first approach. Moreover, it applied backward sub-pattern and backward super-pattern methods in CLOSPAN to prune search space.

Another proposition was proposed by H. Pinto et al.[9]. A multidimensional sequence in this proposition is one or more dimensions of information in which the order of dimension values is not important, alongside a single dimension of information in which order is important. The PSFP and HYBRID algorithms were proposed to mine multidimensional sequences [9]. PSFP mines sequences using the PrefixSpan algorithm [11] followed by mining multidimensional information associated with mined sequences using the FP-growth algorithm [12]. HYBRID mines multidimensional information using the BUC algorithm [18] followed by mining sequences associated with mined multidimensional information using the PrefixSpan algorithm. The performance of PSFP is much faster than that of HYBRID at a low minimum support threshold. When dimensionality is high, the major cost is the mining of multidimensional information. PSFP is still faster than HYBRID because it only mines multidimensional information that occurs with existing sequences. However, HYBRID is better alternative than PSFP when data sets are sparse with respect to dimension values present, and dense with respect to the sequences present. In addition, H. Pinto et al. proposed two approaches for mining multidimensional sequences in [10]. The first approach is integration of efficient frequent sequences mining and multidimensional analysis methods. Two algorithms were developed for the first approach, Dim-Seq and Seq-Dim. Dim-Seq takes the same approach mining of HYBRID, whereas Seq-Dim is similar to PSFP. The second approach is that embed multidimensional information into sequences and mining the whole set using the PrefixSpan algorithm. The Uniseq algorithm was developed for the second approach. The performance study of three algorithms found that Seq-Dim is the fastest algorithm

in the most cases; dimensionality is high, the number of tuples is high and a minimum support threshold is low. In addition, it is the best algorithm when data sets are dense with respect to both dimension value combinations and sequential items. Uniseq is the most effective when the total number of sequential items plus other dimension values is small. Dim-Seq outperforms the other methods when data sets are sparse with respect to dimension value combinations, but dense with respect to the sequences present.

Since the last position generates a large number of redundant multidimensional sequences, closed multidimensional sequence mining [29, 30] was proposed to eliminate redundant multidimensional sequences. An efficient method for mining multidimensional sequences was proposed with two major steps. The first step is a combination of closed itemset mining with closed sequence mining. The CLOSET and CLOSPAN algorithms are exploited in this step for mining closed itemsets and closed sequences, respectively. Since redundant sequences may be produced in the first step, an elimination of the redundant sequences is used in the second step. In first step, experiments were investigated using two integration approaches: mining closed iemsets followed by mining closed sequences and vice versa. The experiments determine factors affecting the number of candidate patterns. From the experiment results, they show that the first approach gives less the number of patterns than the second one on data sets with few dimensions. For data sets with many dimensions, the results are opposite.

Furthermore, the first step in the method was improved to avoid mining unnecessary patterns by using an idea that any patterns found in dataset X must also appear in dataset Y if $X \subseteq Y$ [40]. Therefore, the closed sequences associated with a closed itemset I can be found from the closed sequences associated with a sub-pattern of I when mining closed itemset followed by mining closed sequence mining. We can take the same approach when mining closed sequence followed by mining closed itemsets.

2.2.3 Top-*k* Closed Sequence Mining

Mining top-*k* closed itemsets of length no less than a minimal length was extended to mine top-*k* closed sequences of length no less than a minimal length [31, 32]. Some ideas in top-*k* closed itemset mining were used in top-*k* closed sequence mining. The TSP algorithm was developed for mining top-*k* closed sequences. It exploits multi-pass search space traversal algorithm to find frequent sequences and raise a minimum support threshold dynamically. The raised threshold is used to prune unpromising branches in the search space. Closed sequences

verification method is employed to determine whether a pattern should be added to the top-k result set tree, a sequence is not added to the result set if its support is equal to support of its extension. TSP is further improved by two methods; the first one is exploiting early termination by considering the same projected database to prunes search space which is similar to CLOSPAN, and the second one is using minimum length constraint to create only projected sequence whose length is not less than a minimum length subtracting length of prefix sequence. However, TSP need to store a result set in memory to ensure that there is no sequences in a database that can absorb more than one sequence in the current result set.

CHAPTER 3

Methods

As the chapter 2, the most algorithms for mining top-k itemsets and top-k closed itemsets have to find a final threshold before actual mining phase. Moreover, they need to maintain candidates to check that the candiates are real answers. To solve the problems, this thesis proposes a new efficient algorithm, called TOPK_CLOSED, to mine top-k closed itemsets. The TOPK_CLOSED algorithm generates closed itemsets in descending order of their supports by adopting a best first search strategy. A final threshold is discovered when the k^{th} closed itemset is found. Thus, TOPK_CLOSED is unnecessary to find a final threshold before actual mining phase. Moreover, TOPK_CLOSED directly generates closed itemsets so that it does not need to keep candidates in memory.

Since top-k closed itemset mining cannot answer when users are interested in k most frequent closed itemsets of each length, N-most interesting closed itemset mining is introduced in this thesis to retrieve a desired number of closed itemsets in different length. Like top-k closed itemset mining, N-most interesting mining can retrieve closed itemsets without providing a minimum support threshold of each length. In addition, input parameters, the desired number of closed itemsets of each length (N) and the upper bound of length (l_{max}), are easy for users to control with their applications. An efficient algorithm, NCLOSED, is implemented for mining N-most interesting closed itemsets by adapting the TOPK_CLOSED algorithm.

This chapter is organized as follows. The basic definitions of top-k closed itemset mining and N-most interesting closed itemset mining are given in section 3.1. The methods of the TOPK_CLOSED and NCLOSED algorithms are presented in section 3.2. Since the methods of TOPK_CLOSED and NCLOSED are similar, so they are explained in the same topic. The performance analysis of TOPK_CLOSED and NCLOSED are shown in time complexity and space complexity in section 3.3.

3.1 Basic Definitions

Let $D = \{d_1, d_2, ..., d_m\}$ be a non-empty set of all transaction ids in the database, and $I = \{i_1, i_2, ..., i_n\}$ be a non-empty set of distinct items appearing in the database. A tidset (a set of

transaction ids) $T = \{t_1, t_2, ..., t_h\}$ is a non-empty subset of D, where $h \leq m$. An itemset (a set of items) $X = \{x_1, x_2, ..., x_l\}$ is a non-empty subset of I, where $l \leq n$. In this thesis, an itemset $X = \{x_1, x_2, ..., x_l\}$ is denoted as $X = x_1 x_2 ... x_l$.

Definition 3.1 [17] The support of itemset X is the number of transaction ids containing X that is $|g(X)|$, denoted as $supp(X)$.

Definition 3.2 [17] The length of itemset X is the number of items contained in X. An itemset having length l is denoted as *l-itemset.*

Definition 3.3 [17] Itemset X is a sub-itemset of itemset Y if and only if X is a subset of Y. Y is called a super-itemset of X, denoted as $X \subseteq Y$.

Observation: If $X \subseteq Y$ and $supp(X) = supp(Y)$, X will be absorbed by Y.

Definition 3.4 [26, 27] Given a minimum support threshold min_supp, an itemset X is a frequent closed itemset if $\varsigma(X) = X$ and $supp(X) \geq min\_supp$.

Definition 3.5 [13, 16] The top-k closed itemsets are the set of (frequent) closed itemsets having support $\geq s$, where s is the support of the k^{th} closed itemset in the descending support value list of the closed itemsets.

Example 3.1 Table 3.1 shows a sample itemset database. The characters *a-h* represents items in the database.

Table 3.1 An example database

Tid	Items
1.	*a b c d f g h*
2.	*a b c d f g*
3.	*c f g h*
4.	*b c d f g*
5.	*a c f*
6.	*c f*
7.	*a b c f g h*
8.	*a b c h*
9.	*a h*

Given $k = 8$, the 8 itemsets having highest eight supports are *c:*8, *fc:*7, *a:*6, *h:*5, *ac:*5, *bc:*5, *gcf:*5, and *afc:*4, where the number after : is the support of the itemset. Since there is more than one closed itemset having the same support, closed itemsets having the same support of the 8th closed itemset are included in the set of resulting itemsets. Thus, the set of resulting itemsets is { *c:*8, *fc:*7, *a:*6, *h:*5, *ac:*5, *bc:*5, *gcf:*5, *afc:*4, *bca:*4}.

Definition 3.6 The *N*-most interesting *l*-closed itemsets are the set of *l*-closed itemsets having support $\geq s$, where s is the support of the N^{th} *l*-closed itemset in the descending support value list of the *l*-closed itemsets.

Definition 3.7 The *N*-most interesting closed itemsets are the union of the *N*-most interesting *l*-closed itemsets for each $1 \leq l \leq l_{\max}$, where $l_{\max}$ is the upper bound of the desired length of closed itemsets.

Example 3.2 Given $N = 3$, $l_{max} = 3$, the set of 3 1-closed itemsets is {*c:*8, *a:*6, *h:*5} that is the three closed itemsets having length 1 with the highest three supports. The set of 3 2-closed itemsets is {*fc:*7, *ac:*5, *bc:*5}, and the set of 3 3-closed itemsets is {*gcf:*5, *afc:*4, *bca:*4}. The combination of the three sets is the set of resulting 3-most interesting closed itemsets{*c:*8, *a:*6, *h:*5, *fc:*7, *ac:*5, *bc:*5, *gcf:*5, *afc:*4, *bca:*4}.

3.2 Method Development

3.2.1 Closed Itemset Generation

Both the TOPK_CLOSED and NCLOSED algorithms generate closed itemsets without keeping candidates by computing the closure of itemsets based on lemma 3.1.

Lemma 3.1 [4, 5] Given an itemset X and an item $i \in I, g(X) \subseteq g(\{i\})$ if and only if $i \in \varsigma(X)$.

Proof Case 1: $g(X) \subseteq g(\{i\}) \rightarrow i \in \varsigma(X)$

Since $g(X) \subseteq g(\{i\})$, it show that $\{i\} \subseteq X$ and $X \cup \{i\} = X$. Thus, $g(X \cup \{i\}) = g(X)$.

If $g(X \cup \{i\}) = g(X)$, then $f(g(X \cup \{i\})) = f(g(X))$ and $\varsigma(X \cup \{i\}) = \varsigma(X)$.

Therefore, $i \in \varsigma(X)$.

Case 2: $i \in \varsigma(X) \rightarrow g(X) \subseteq g(\{i\})$

If $i \in \varsigma(X)$, then $g(X) = g(X \cup \{i\})$.

Since $g(X \cup \{i\}) = g(X) \cap g(\{i\}), g(X) = g(X) \cap g(\{i\})$ holds too.

Thus, $g(X) \subseteq g(\{i\})$.

From case 1 and 2, we can conclude that $g(X) \subseteq g(\{i\}) \leftrightarrow i \in \varsigma(X)$. □

Example 3.3 A tidset containing itemset *fg* is $g(\{fg\}) = \{1, 2, 3, 4, 7\}$ and A tidset containing item *c* is $g(\{c\}) = \{1, 2, 3, 4, 5, 6, 7, 8\}$. We can see that $g(\{fg\}) \subseteq g(\{c\})$. Therefore, $c \in \varsigma(\{fg\})$.

With lemma 3.1, we can calculate the closure of itemset X by including any items having a superset of tidset of X. From this approach, the same closure may be calculated twice from two different itemsets, so the same closed itemset is produced twice. For example, closed itemset *bcfg* is calculated twice from itemset *bcf* and *gcf*. To avoid this problem, the itemsets have to be checked for duplication. Itemsets with non-duplicate are used to find closed itemsets, whereas itemsets with duplicate are discarded, because they lead to redundant of computation of the same closed itemsets.

Definition 3.8 [4, 5] Let $X = Y \cup \{i\}$ be an itemset, where Y is a closed itemset, $i \in I$ and $i \notin Y$. X is said to be a non-duplicate generator if and only if $\varsigma(X) = X$.

Definition 3.9 Let $X = Y \cup \{i\}$ be an itemset, where Y is a closed itemset, $i \in I$ and $i \notin Y$. The set of post-items of X is defined as follows.

$$post - items(X) = \{r \mid r \in I, r \notin X, and\ i \prec r\} \tag{3.1}$$

Observation: $i \prec r$ in our algorithm means that i appears before r in sorted items list. The sorted items list consists of distinct items that are sorted by their supports in descending order.

Theorem 3.1 Let $X = Y \cup \{i\}$ be an itemset, where Y is a closed itemset, $i \in I$ and $i \notin Y$. If $\exists r \in post - items(X)$ such that $g(X) \subseteq g(\{r\})$, then X is not a non-duplicated generator.

Proof If $g(X) \subseteq g(\{r\})$, then $r \in \varsigma(X)$ according to lemma 3.1. Since $r \in post - items(X), r \notin X$ according to definition 3.9. $r \in \varsigma(X)$ but $r \notin X$. So $\varsigma(X) \neq X$. Based on definition 3.8, X is not a non-duplicated generator. □

From theorem 3.1, we can check that itemset X is a non-duplicated generator or not by checking tidset of X with tidset of post-items of X. Since distinct items are sorted by their supports in descending order, some post-items are not used for duplicate checks if their supports are less than the support of X. If itemset X is not a non-duplicated generator, it will be discarded. Otherwise, its closure is computed to find a closed itemset based on theorem 3.2.

Definition 3.10 Let $X = Y \cup \{i\}$ be an itemset, where Y is a closed itemset, $i \in I$ and $i \notin Y$. The set of pre-items of X is defined as follows.

$$pre - items(X) = \{j \mid j \in I, j \notin X, and\ j \prec i\} \tag{3.2}$$

Theorem 3.2 $X = Y \cup \{i\}$ be an non-duplicated generator, where Y is a closed itemset, $i \in I$ and $i \notin Y$. If $j \in pre-items(X)$ and $g(X) \subseteq g(j)$, then $j \in \varsigma(X)$.

Proof X is a non-duplicated generator, it shows that there is no $j \in post-items(X)$ such that $g(X) \subseteq g(\{j\})$ according to theorem 3.1. Therefore, only $j \in pre-items(X)$ are considered to compute a closure. If $g(X) \subseteq g(j)$, then $j \in \varsigma(X)$ according to lemma 3.1. □

Theorem 3.2 shows how to calculate a closure in our algorithm. Only pre-items of itemset X is considered to calculate the closure of X as shown in example 3.4.

Example 3.4 From table 3.1, the sorted items list consists of *c, f, a, b, g, h* and *d*. Considering item *b*, post-items of *b* are *g h* and *d*. Item *b* is checked for duplicate before calculating its closure. Since $g(b) \not\subset g(g), g(b) \not\subset g(h)$ and $g(b) \not\subset g(d)$, b is a non-duplicated generator and $\varsigma(b)$ is then calculated. Pre-items of b are *c*, *f* and *a*. Since $g(b) \subset g(c)$ but $g(b) \not\subset g(f)$ and $g(b) \not\subset g(a)$, only *c* is included in $\varsigma(b)$. Thus, the closure of *b* is *bc* that is a closed itemset. A closed itemset is expanded with remaining pre-items that are not included in it. Closed itemset *bc* is expanded with the remaining pre-items, *f* and *a*. We get two extended itemsets $bcf = bc \cup f$ and $bca = bc \cup a$. Considering itemset *bcf*, post-items of *bcf* are *a, g, h* and *d*. Since $g(bcf) \subset g(g)$, *bcf* is not a non-duplicated generator. *bcf* is discarded, because it and any itemsets produced from it are absorbed by itemsets generated from item *g*. From this approach, we can reduce search space to find closed itemsets.

3.2.2 Finding Closed Itemsets with High Supports

The TOPK_CLOSED and NCLOSED algorithms avoid the phase for finding a final minimum support threshold before mining phase. They can generate closed itemsets in descending order of their supports by adopting best first search. We use a priority queue to implement best first search because it can schedule highest-priority-first order. Since the support of an itemset is the same as the support of its closure as shown in lemma 3.2, it is exploited as priority in queue. An itemset with the highest support in queue is firstly accessed, because it can lead to a closed itemset having higher support than other itemsets in queue according to theorem 3.3.

Lemma 3.2 The support of an itemset X is equal to the support of the closure of X.

Proof Suppose X is an itemset. We know that $\varsigma(X) = f(g(X))$, then $g(\varsigma(X)) = g(f(g(X)))$. But $g(f(g(X))) = \varsigma'(g(X)) = g(X)$ based on the property 2.7 of Galois connection (see section

2.1.1), then $g(\varsigma(X)) = g(X)$. So $|g(\varsigma(X))| = |g(X)|$. Therefore, $supp(\varsigma(X)) = supp(X)$. □

Theorem 3.3 For itemsets X and Y, if $supp(X) > supp(Y)$, then $supp(\varsigma(X)) > supp(\varsigma(Y))$.

Proof We know that $supp(X) = supp(\varsigma(X))$ and $supp(Y) = supp(\varsigma(Y))$ according to lemma 3.2. Since $supp(X) > supp(Y)$, $supp(\varsigma(X)) > supp(\varsigma(Y))$. □

A known data structure for maintaining priority queue is a binary heap [34]. In our algorithm, we determine an array as a binary heap tree as shown in figure 3.1. Each node in the tree contains four fields; *itemset*, *tids*, *active* and *capacity*.

- The *itemset* field holds an itemset X.
- The *tids* field holds tidset containing X.
- The *active* field is a Boolean which is set to true if the node contains an itemset and tidset to which it belong. It is set to false if the node is empty.
- The *capacity* field contains the number of inactive nodes in the sub-tree rooted to current node.

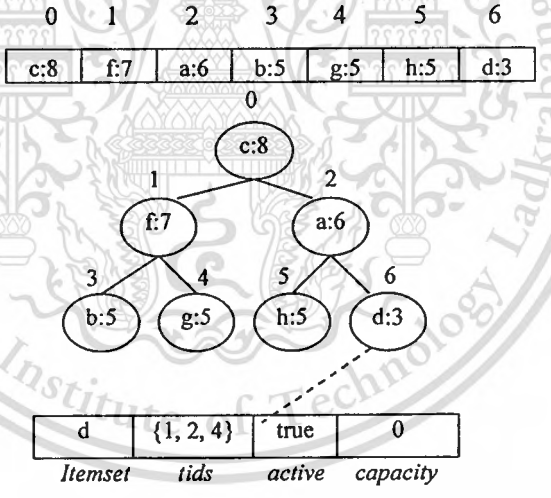


Figure 3.1 A binary heap structure for TOPK_CLOSED and NCLOSED.

Based on heap concept, the support of parent node is always greater than or equal to those of children nodes. Thus, a new itemset is considered its support to be inserted in queue by using en-queue operation as shown in figure 3.2(a). The highest support is always in the root node. It is always removed from queue by using de-queue operation as shown in figure 3.2(b). Therefore, an itemset having highest support in queue is always accessed to find a closed itemset. As a result, closed itemsets are found in descending order of their supports.

Subroutine: en-queue (v)
Input: queue q and an itemset v
Output: none
Method:
1. if q is full then
2. extend q and increate capacity of root node
3. i=0
4. while(1)
5. if $q[i].active$ = = false then
6. insert v in $q[i]$
7. break
8. else if $supp(v) > supp(q[i])$ then
9. swap v and q[i]
10. decrease q[i].*capacity*
11. if $left(i).capacity > 0$ then
12. $i = left(i)$
13. else
14. $i = right(i)$

(a)

Subroutine: de-queue ()
Input: queue q
Output: an itemset with highest support v in q
Method:
1. move $q[0]$ to v
2. $q[0].active$ = false
3. increase $q[0].capacity$
4. $i = 0$
5. while(1)
6. if $left(i).active$ and $right(i)$.active == false then
7. break
8. find the node $q[k]$ with the highest support among $left(i)$ and $right(i)$
9. $q[i].active$ = true
10. move $q[k]$ to $q[i]$
11. $q[k].active$ = false
12. increate $q[k].capacity$
13. $i = k$
14. return v

(b)

Figure 3.2 (a) en-queue subroutine and (b) de-queue subroutine.

3.2.3 Finding the Final Threshold and Stopping the Mining Process

Most previous works of top-k itemset mining, top-k closed itemset mining and N-most interesting itemset mining find a final minimum support threshold before mining phase, but our algorithms does not need to find the final threshold before mining phase. Our algorithm can produce closed itemsets in descending order of their supports so that the final threshold is discovered when k^{th} closed itemset or N^{th} closed itemset is found.

For the TOPK_CLOSED algorithm, the mining process will be stopped if an itemset having lower support than the final threshold is accessed in queue, as shown in theorem 3.4.

Theorem 3.4 If the support of itemset X is less than the support of k^{th} closed itemset, then $\varsigma(X)$ is not a top-k closed itemset.

Proof Let X be an itemset, s be the support of k^{th} closed itemset and $supp(X) < s$. Based on lemma 3.2, $supp(\varsigma(X)) = supp(X)$, then $supp(\varsigma(X)) < s$. Therefore, $\varsigma(X)$ is not a top-k closed itemset according to definition 3.5. □

For the NCLOSED algorithm, it exploits a simple structure, arrays, to find the final threshold of each length. Two arrays are used in this algorithm. The first one, array C, is used for counting the number of closed itemsets of each length. The second one, array S, is exploited for storing the final thresholds. The length of both arrays is *N*. Indices in the arrays correspond to the length of closed itemsets. Firstly, all elements in both arrays are initialized to zero. If a closed itemset is found, an element indexed by length of the closed itemset - 1 in C is increased by 1. After the element is increased up to *N*, an element indexed by length of the closed itemset - 1 in S is set to the support of the N^{th} closed itemset. For example, suppose N =3 and l_{max} = 3 are given to find 3-most interesting closed itemsets. In figure 3.3, after closed itemse *bc* is found, element in C indexed by 1 (the length of *bc* - 1) is increased from 2 to 3 which is *N*. Thus, the support of *bc* is used as the final threshold of 2-closed itemsets and stored in element indexed by 1 in S.

	Array C			Array S		
	0	1	2	0	1	2
c:8	1	0	0	0	0	0
fc:7	1	1	0	0	0	0
a:6	2	1	0	0	0	0
ac:5	2	2	0	0	0	0
bc:5	2	3	0	0	5	0
gcf:5	2	3	1	0	5	0
h:5	3	3	1	5	5	0
afc:4	3	3	2	5	5	0
bca:4	3	3	3	5	5	4

Figure 3.3 Finding final thresholds of 3-most interesting closed itemsets generation.

The minimum value of the final thresholds is used to stop the mining process. If an itemset having lower support than the minimum value is found, the mining process of NCLOSED will be stopped as shown in theorem 3.5.

Theorem 3.5 If the support of itemset X is less than the minimum value among the supports of the N^{th} l-closed itemset for $1 \le l \le l_{max}$, then X is not an N-most interesting closed itemset.

Proof Let X be an itemset, s be the minimum value among the support of the N^{th} l-closed itemsets for $1 \le l \le l_{max}$ and $supp(X) < s$. Based on lemma 3.2, $supp(\varsigma(X)) = supp(X)$, then

$supp(\varsigma(X)) < s$. Therefore, $\varsigma(X)$ is not any N-most interesting l-closed itemsets according to definition 3.6. We can conclude that $\varsigma(X)$ is not N-most interesting closed itemsets according to definition 3.7. □

3.2.4 The Proposed Algorithms

This section includes all strategies of the TOPK_CLOSED and NCLOSED algorithms. and shows the proposed algorithms in figure 3.5 and 3.6, respectively.

For TOPK_CLOSED, the pseudo code is shown in figure 3.5. TOPK_CLOSED first scans a database to find a tidset of each distinct item (line 1). The distinct items are sorted in descending order of their supports and added in queue (lines 2-3).The final threshold is initialized to 1 (line 4). The topmost itemset X is removed from queue (line 6). If the support of X is not less than the final threshold (line 7), it is firstly checked whether it is a non-duplicated generator or not as shown in figure 3.4(a) (line 8). If it is a non-duplicated generator, it will be computed to find a closed itemset as shown in figure 3.4(b) (line 9). The number of closed itemsets found is counted (line 11). After it tills k, the final threshold is reset to the support of the k^{th} closed itemset (lines 12-13). Each of closed itemset is expanded to new itemsets and the new itemsets are inserted in queue (lines 14-15). Finally, the same process will be repeated until the support of the topmost itemset in queue is lower than the final threshold. The mining process is then stopped and the resulting patterns are returned.

Subroutine: Generator_check(X)
Input: An itemset X
Output: If X is a non-duplicate generator, 1 is returned. Otherwise, 0 is returned.
Method:
1. while($supp(X) == supp(l_i)$) ; $l_i \in post\text{-}items(X)$
2. if ($g(X) \subseteq g(l_i)$) then
3. return 0
4. return 1

(a)

Subroutine: Closure_calculate(X)
Input: A itemset X
Output: A Closure of X, Y
Method:
1. $Y = X$
2. while($supp(X) == supp(j_j)$) ; $j_j \in pre\text{-}items(X)$
3. if ($g(X) \subseteq g(j_j)$) then
4. $Y = Y \cup j_j$
5. return Y

(b)

Figure 3.4 (a) Duplicate check subroutine and (b) Closure calculation subroutine.

Algorithm: TOPK_CLOSED

Input: A database D, the desired number of closed itemsets k.

Output: A set of top-k closed itemset TCI.

Method:

1. scan database D and tidset of each item
2. sort distinct items in descending order of their supports
3. insert the distinct items in queue
4. $suppK = 1$
5. while(1)
6. X = en-queue()
7. if $supp(X) < suppK$ then break
8. if !Generator_check(X) then
9. C = Closure_calculate(X)
10. $TCI = TCI \cup X$
11. $m = m+1$
12. if$(m==k)$
13. $suppK$ = the support of k^{th} top-k closed itemset
14. $P = \{Y \cup i_1, Y \cup i_2, ..., Y \cup i_n\}$; $i \in$ *pre-items(Y)*
15. en-queue(P)

Figure 3.5 The TOPK_CLOSED algorithm.

For the NCLOSED algorithm, its pseudo code is shown in figure 3.6. Like the TOPK_CLOSED algorithm, NCLOSED first scans a database to find a tidset of each distinct item (line 1). The distinct items are sorted in descending order of their supports and inserted in queue (lines 2-3). Unlike TOPK_CLOSED, NCLOSED use two arrays to find final threshold of each length. Elements in two arrays, C, S are initialized to 0 (lines 4-5). Queue is checked is empty or not (line 7). If queue is not empty, the topmost itemset X is removed from queue (line 8). X is firstly checked whether it is a non-duplicated generator or not as shown in figure 3.4(a) (line 9). If it is a non-duplicated generator, it will be computed to find a closed itemset as shown in figure 3.4(b) (line 10). Next, the support of the closed itemset is compared with the final threshold of its length in S. If it is not less than the final threshold, it is included in the set of N-most interesting closed itemsets (lines 12-13). The final threshold of l-closed itemsets is finally set when N^{th} l-closed itemset is found (lines 11-12). Then the closed itemset is checked whether its length is not less than l_{max} or not (line 16). If it is less than l_{max}, the closed itemset is expanded with pre-items (line 17). All extended itemsets are inserted in queue (line 18). The same process will be repeated

until the support of topmost itemset in queue is lower than the minimum value among the final thresholds. The mining process is then stopped and the resulting patterns are returned.

Algorithm: NCLOSED
Input: A database D, the upper bound of the length l_{max} and the desired number of l-itemsets N.
Output: N-most interesting closed itemsets NCI.
Method:

```
1.  scan database D and find tidset in each item
2.  sort distinct items in descending order of their supports
3.  insert the distinct items in queue q
4.  S[0] = S[1] =... = S[l_max-1] = 0
5.  C[0] = C[1] =... = C[l_max-1] = 0
6.   while(1)
7.       if queue is empty then break
8.       X = de-queue()
9.       if !Generator_check(X) then
10.          Y = Closure_calculate(X)
11.          if supp(Y) ≥ S[Y.len] then
12.              NCI = NCI ∪ Y
13.          C[Y.len] =C[Y.len]+1
14.          if C[Y.len] ==N then
15.              S[Y.len] = supp(Y)
16.          if Y.len < l_max then
17.              P = {Y∪i1, Y∪i2,..., Y∪in} ; i ∈ pre-items(Y)  //Extend Y
18.                  en-queue(P)
```

Figure 3.6 The NCLOSED algorithm.

3.3 Performance Analysis

In this section, performances of the TOPK_CLOSED and NCLOSED algorithms are analyzed in time complexity and space complexity as follows.

3.3.1 The TOPK_CLOSED Algorithm

3.3.1.1 Time Complexity

The TOPK_CLOSED aglrithm first scans the database to find a tidset of each distinct item. The worst case cost of the scan is at most the number of transaction ids in the database, $O(|D|)$. TOPK_CLOSED removes the itemset having highest support in queue to find a closed itemset by using de-queue operation. Then new itemsets are expanded from the closed itemset and they are inserted in queue using en-queue operation. The number of new itemsets

expanded from a closed itemset is at most the number of distinct items $|I|-1$. We have to remove k itemsets from queue to find top-k closed itemsets, so the total number of itemsets inserted in queue is at most $k(|I|-1)$. The worst case time complexity for removing and inserting itemsets in queue are $O(k\log(k(|I|-1)))$ and $O(k(|I|-1)\log(k(|I|-1)))$, respectively. After removing the itemset X with highest support from queue, X is checked for duplicate. The worst case time complexity of a duplicate check happens when X is a non-duplicated generator. In such case, tidsets of all post-items will be checked with $g(X)$. Every tidsets of the post-items is compared to be included in $g(X)$. A tidset of each post-item is at most the number of transaction ids in the database, $|D|$. The number of post-items is at most as big as $|I|-1$. Therefore, the worse case time complexity of the duplicate check in our algorithm is $O(|D|(|I|-1))$. After checking for duplication, X is computed to find its closure. The worst case time complexity of a closure computation occurs when $\forall j \in pre-items$ such that $g(X) \subseteq g(j)$. The number of pre-items is at most $|I|-1$. Every tidsets of the pre-items is compared to be included in $g(X)$. A tidset of each pre-item is at most the number of transaction ids in the database, $|D|$. Therefore, the worse case time complexity of closure computation in our algorithm is $O(|D|(|I|-1))$. In our algorithm, distinct items used as post-items to check duplicates are not used as pre-items to compute closure, and distinct items used as pre-items to compute closure are not used as post-items to check duplicates. Thus, the worse case time complexity of duplicated check and closure computation in our algorithm is $O(|D|(|I|-1))$. In conclusion, the worse case time complexity of the TOPK_CLOSED algorithm is $O(|D||I|+k\log(k(|I|-1))+k(|I|-1)\log(k(|I|-1)))$.

3.3.1.2 Space Complexity

The worst case memory occupation of the TOPK_CLOSED algorithm occurs when $\varsigma(X)=X$ and all itemsets are non-duplicated generators. For each closed itemset, it can expand the number of new itemsets at most the number of distinct items $|I|$-1. We have to expand new itemsets to insert in queue from k closed itemsets. Thus, the number of itemsets in queue is at most $k(|I|-1)$. Each itemset keeps tidset for duplicate check and closure computation. A tidset of each itemset is at most the total number of transaction ids in the database, $|D|$. In conclusion, the worst case space complexity of the TOPK_CLOSED algorithm is $O(|D|k(|I|-1))$.

3.3.2 The NCLOSED Algorithm

3.3.2.1 Time Complexity

The NCLOSED algorithm first scans the database to find a tidset of each distinct item. The worst case cost of the scan is at most the number of transaction ids in the database, $O(|D|)$. Our algorithm removes the itemset having highest support in queue to find a closed itemset by using de-queue operation. Then new itemsets are expanded from the closed itemset and they are inserted in queue using en-queue operation. The number of new itemsets expanded from a closed itemset is at most the number of distinct items $|I|-1$. We have to remove $Nl_{\max}$ itemsets from queue to find N-most interesting closed itemsets, where N is the desire number of closed itemsets of each length and $l_{\max}$ is the upper bound of the desired length of itemsets. Therefore, the total number of itemsets inserted in queue is at most $Nl_{\max}(|I|-1)$ and each insert takes at most $\log(Nl_{\max}(|I|-1))$. To find an N-most interesting closed itemset, the topmost itemset in queue is removed by taking at most $\log(Nl_{\max}(|I|-1))$. Thus, the worst case time complexity for inserting and removing itemsets to find all resulting patterns are $O(Nl_{\max}(|I|-1)\log(Nl_{\max}(|I|-1)))$ and $O(Nl_{\max}\log(Nl_{\max}(|I|-1)))$, respectively. After removing the itemset X with highest support from queue, X is first checked for duplication. The worst case time complexity of a duplicate check happens when X is a non-duplicated generator. In such case, tidsets of all post-items will be checked with $g(X)$. Every tidsets of the post-items is compared to be included in $g(X)$. A tidset of each post-item is at most the number of transaction ids in the database, $|D|$. The number of post-items is at most as big as $|I|-1$. Therefore, the worse case time complexity of the duplicate check in our algorithm is $O(|D|(|I|-1))$. After checking for duplication, X is computed to find its closure. The worst case time complexity of a closure computation occurs when $\forall j \in pre-items$ such that $g(X) \subseteq g(j)$. The number of pre-items is at most $|I|-1$. Every tidsets of the pre-items is compared to be included in $g(X)$. A tidset of each pre-item is at most the number of transaction ids in the database, $|D|$. Therefore, the worse case time complexity of closure computation in our algorithm is $O(|D|(|I|-1))$. In the NCLOSED algorithm, distinct items used as post-items to check duplicates are not used as pre-items to compute closure, and distinct items used as pre-items to compute closure are not used as post-items to check duplicates. Therefore, the worse case time complexity of duplicated check and closure computation in our algorithm is $O(|D|(|I|-1))$. In conclusion, the worse case time complexity of the NCLOSED algorithm is $O(|D||I|+Nl_{\max}\log(Nl_{\max}(|I|-1))+Nl_{\max}(|I|-1)\log(Nl_{\max}(|I|-1)))$.

3.3.2.2 Space Complexity

The worst case memory occupation of the NCLOSED algorithm occurs when $\varsigma(X) = X$ and all itemsets are non-duplicated generators. For each closed itemset, it can expand the number of new itemsets at most the number of distinct items $|I|$-1. We have to expand new itemsets to insert in queue from $Nl_{\max}$ closed itemsets. Thus, the number of itemsets in queue is at most $Nl_{\max}(|I|-1)$. Each itemset keeps tidset for duplicate check and closure computation. A tidset of each itemset is at most the total number of transaction ids in the database, $|D|$. In conclusion, the worst case space complexity of the NCLOSED algorithm is $O(|D|Nl_{\max}(|I|-1))$.

CHAPTER 4

Experimental Evaluation

In this section, we report our experimental studies of the TOPK_CLOSED and NCLOSED algorithms. All experiments were performed on Intel(R) Core (TM) 2 CPU 1.66 GHZ with 504 MB of RAM, running Microsoft Windows XP Professional version 2002. Our algorithms were coded in C++. All algorithms were complied and run on the Cygwin program.

4.1 Data Sets

Both real and synthetic data sets are used in our experiments. They are grouped into two categories; dense and sparse data sets. Dense data sets consist of mushroom, connect, chess and pumsb. The mushroom data set contains characteristic of various species of mushrooms. The connect and chess data sets are derived from game state information. The pumsb data set contains census data. All dense data sets contain many long frequent closed itemsets. For sparse data sets, they are gazelle and T10I4D8k. The gazelle data set comes from click stream data from a small dot-com company called Gazelle.com. A portion of this data set was used in KDD-Cup 2000 competition. The T10I4D8k is a synthetic data set generated from IBM generator. The mushroom, connect, chess, pumsb and gazelle are real data sets. They are publicly available from the FIMI repository page http://fimi.cs.helsinki.fi/data/. Table 4.1 presents characteristic of the real and synthetic data sets used in our experiments. It shows type of data sets, the number of items, the average transaction length and the number of transactions in each data set.

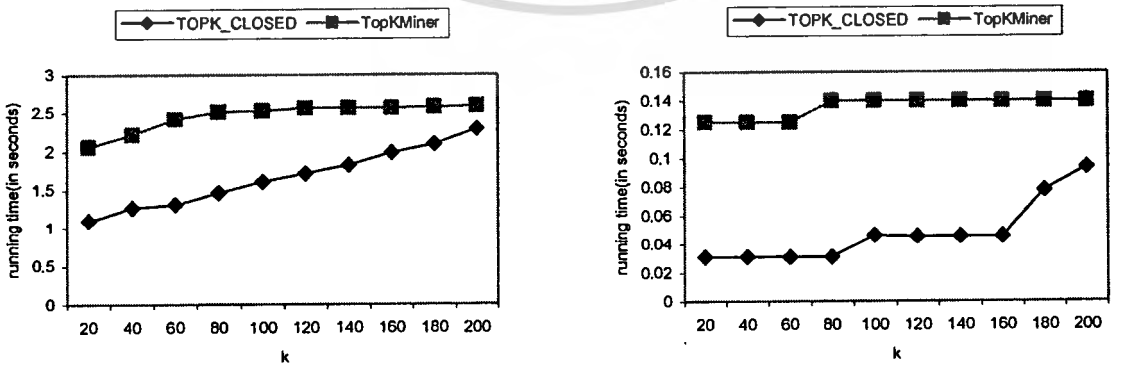
Table 4.1 Data sets characteristics

Data set	Type	#Items	Avg. Length	#Transactions
mushroom	Dense	119	23	8,124
connect	Dense	130	43	67,557
chess	Dense	75	37	3,196
pumsb	Dense	2,113	74	49,046
gazelle	Sparse	497	2.5	59,601
T10I4D8k	Sparse	862	10	8,000

4.2 Experimental Results

For experimental study of the TOPK_CLOSED algorithm, we investigate experiments by comparing the efficiency of it with a well-know algorithm, TopKMiner. The TopKMiner algorithm was coded by C++ and the original source code is provided to us by its authors. We did not include the TFP algorithm in our experiments because TopKminer was already proven that it outperforms TFP [1].

Figure 4.1, 4.2, and 4.3 show the running time of TOPK_CLOSED and TopKMiner on different data sets for k ranging from 20 to 200. Figure 4.1 (a) shows the running time on the connect data set. We observe that TOPK_CLOSED outperforms TopKMiner. For figure 4.1 (b), it shows the running time of the two algorithms on the chess data set. The performance of TOPK_CLOSED still outperforms TopKMiner. For figure 4.2 (a), it shows the running time of the two algorithms on the mushroom data set. We can see that TOPK_CLOSED starts to outperform TopKMiner, but TopKMiner outperforms TOPK_CLOSED when k is larger than 140. Figure 4.2 (b) shows the running time of the two algorithms on the pumsb data set. It is shown that TopKMiner outperforms TOPK_CLOSED. From figures 4.1 and 4.2, we can see that the running time of TOPK_CLOSED increases almost linearly with k values, whereas the running time of TopKMiner is almost stable when k value is increased. Thus, the running time of TopKMiner outperforms TOPK_CLOSED when k is large. Figure 4.3 shows the running time of the two algorithms on the sparse data sets. For figure 4.3 (a), it shows that the running time of TopKMiner outperforms TOPK_CLOSED on the gazelle data set. Figure 4.3 (b) shows that the running time of ToKMiner also outperforms TOPK_CLOSED on the T10I4D8k.



(a) (b)

Figure 4.1 Performance on (a) connect and (b) chess.

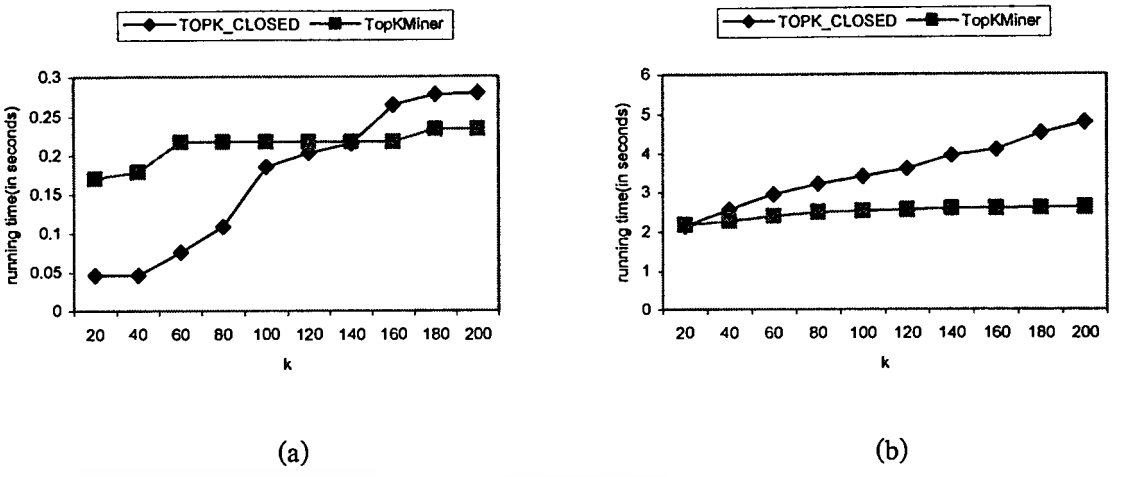


Figure 4.2 Performance on (a) mushroom and (b) pumsb.

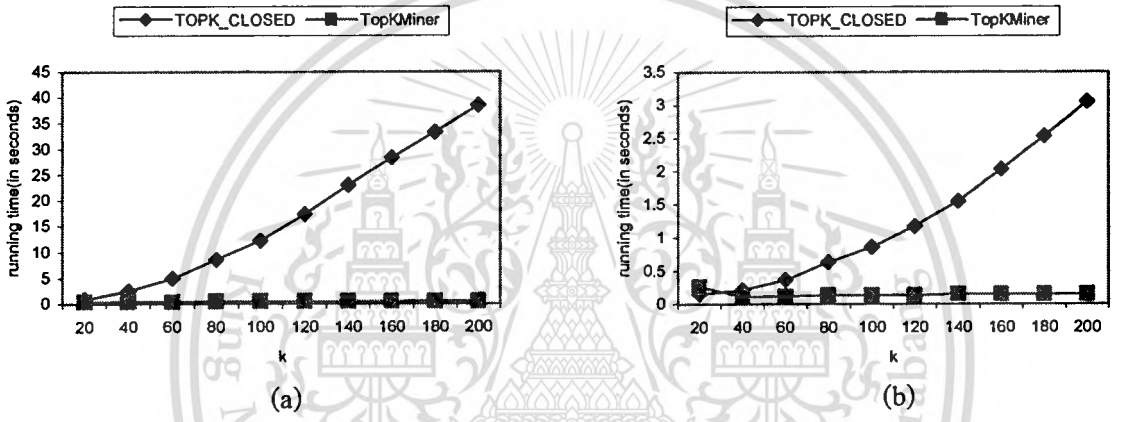


Figure 4.3 Performance on (a) gazelle and (b) T10I4D8k.

From above experimental studies, we conclude that the TOPK_CLOSED algorithm has good performance on dense data sets when pushing the small *k* values into the mining process. It outperforms TopKMiner in the most cases. In these cases, because of TOPK_CLOSED directly performs the actual mining phase, whereas TopKMiner has to create a Patricia trie before actual mining phase. However, when pushing the large *k* values in mining process, the Patricia trie makes TopKMiner efficient in the actual mining phase while TOPK_CLOSED takes a large cost for en-queue and de-queue operators. Moreover, the performance of TOPK_CLOSED is dropped down when running on data sets contained many distinct items such as pumsb and the sparse data sets, because it has expensive cost for duplicate check and closure calculation.

For the NCLOSED algorithm, it is proposed for mining *N*-most interesting closed itemsets, which is first studied in this thesis. Therefore, we design to investigate experiments on a variety of data sets with different input parameters values. The experiments are run on dense and

sparse data sets with different sizes. For dense data sets, mushroom is used as a dense data set with small size while connect is used as a dense data set with large size. For sparse data sets, T10I4D8K is used as a sparse data set with small size while gazelle is used as a sparse data set with large size. Figure 4.4 (a) shows the running time of NCLOSED when l_{max} fixed at 5 and N ranging from 10 to 60. We observe that the running time of NCLOSED on the mushroom data set remains stable over the range of N. For the T10I4D8k data set, the running time increases almost linearly with N. Figure 4.4(b) shows the running time of NCLOSED when N set to 20 and l_{max} ranging from 1 to 11. We can see that the running time on the mushroom and T10I4D8k data sets grows linearly with l_{max}. For figure 4.5 (a), we fix l_{max} at 5 and N ranging from 10 to 60. We observe that the running time on the connect data set increases linearly until $N = 30$ and then it remains stable. For the gazelle data set, the running time of NCLOSED increases almost linearly with N. For figure 4.5 (b), N is set to 20 and l_{max} is ranged from 1 to 11. The running time grows linearly with l_{max} on the connect and gazelle data sets.

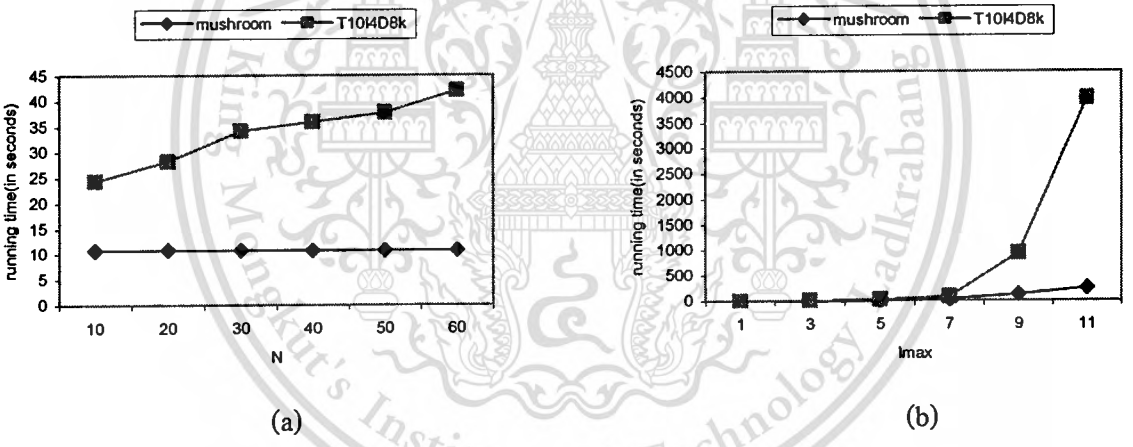


Figure 4.4 Performance on mushroom and T10I4D8K (a) $l_{max} = 5$ (b) $N = 20$

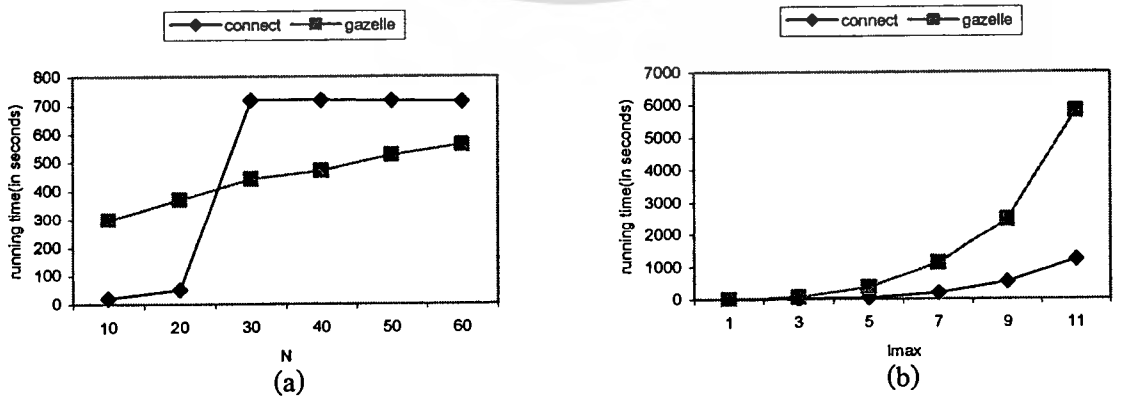


Figure 4.5 Performance on connect and gazelle (a) $l_{max} = 5$ (b) N=20

From above experimental study of the NCLOSED algorithm, we conclude that NCLOSED has good performance on dense data sets when pushing the N constraint deeply into the mining process. This is because dense data sets contain long frequent itemsets. When pushing the N constraint deeply, itemsets in queue always lead to answers. On the other hand, when pushing the l_{max} constraint deeply in the mining process, itemsets from queue may not lead to the answers, but they are required to reach the answers. This is because NCLOSED has to climb from itemsets having length 1 to itemsets having the desired length.

CHAPTER 5

Conclusion and Recommendation

5.1 Conclusion

This thesis studies the most frequent closed itemsets mining. Top-k closed itemset mining and N-most interesting closed itemset mining were proposed to retrieve the desired number of the most frequent closed itemsets without giving a minimum support threshold. They give convenience for users to retrieve the most interesting patterns. Top-k closed itemset mining and N-most interesting closed itemset mining avoid the difficulty of specification of an appropriate threshold. In addition, parameters in the both mining tasks are easily specified by users having no knowledge of mining query and task-specific data.

In this study, we propose two efficient algorithms for mining top-k closed itemsets and N-most interesting closed itemsets. The TOPK_CLOSED algorithm is proposed for mining top-k closed itemsets, and the NCLOSED algorithm is developed for mining N-most interesting closed itemsets. A best first search strategy is used to brown closed itemsets. An itemset with highest support is accessed to find a closed itemset first, so closed itemsets are produced in descending order of their supports. Therefore, the phase for finding a final threshold in our algorithms is unnecessary. The final threshold is found when the k^{th} closed itemset or N^{th} closed itemset is found. After the final threshold is found, it is used to rapidly stop the mining process. If an itemset having lower support than the final threshold is accessed, the mining process will be stopped. As a result, unpromising itemsets are not used in the mining process. Moreover, the proposed algorithms generate closed itemsets without candidates. They directly generate closed itemsets by computing closure of itemsets. An itemset is checked for duplicate before computing its closure to avoid the redundant computation of the same closed itemset. If the set of transaction ids of it is subset of those of any post items, it is discarded. Otherwise, its closure is computed to find a closed itemset by unifying it with the preceding items whose transaction ids include transaction ids of it. After discovering a closed itemset, the closed itemset is grown by extending it with preceding items which are not included in the closed itemset to find other closed itemsets.

In this thesis, we also study the performance of the TOPK_CLOSED and NCLOSED algorithms on different data sets with different parameters values. For TOPK_CLOSED, we

compare the efficiency of it against the TopKMiner algorithm. We found that TOPK_CLOSED outperforms TopKMiner on dense data sets in the most case with small k. For NCLOSED, it performs especially well on dense data sets when pushing the N constraint deeply into the mining process.

5.2 Recommendation

The proposed algorithms generate closed itemsets by closure climbing and growing up closed itemsets with items. For mining N-most interesting closed itemsets, the closure climbing may lead to produce closed itemsets having the undesired length, but the closed itemsets are needed to reach closed itemsets having the desired length. Therefore, an optimized algorithm should be improved to directly produce closed itemsets with the desired length without climbing. Moreover, some of itemsets are unnecessarily inserted in queue, because they are never accessed to find answers. Thus, the unnecessary itemsets should be eliminated before inserting in queue. Furthermore, a good structure may be used to represent data sets before mining phase to reduce computation cost of closure and duplicate check such as CFI-tree, Patricia trie. In addition, our algorithms may be improved to mine top-k closed sequences or N-most interesting closed sequences, but they need to consider the order of itemsets in each sequence. Finally, the proposed algorithms should be improved to have good performance on sparse data sets as well.

REFERENCES

[1] A. Pietracaprina and F.Vandin, "Efficient Incremental Mining of Top-k Frequent Closed Itemsets", **In Proc. of 10th International Conference on Discovery Science**, pp. 275-280, Sendai, Japan, 1-4 October, 2007.

[2] A.W. –C. Fu, R. W. – W. Kwong, and J. Tang, "Mining N-most interesting Itemsets", **In Proc. of 12th Symposium on Methodologies for Intelligent System (ISMIS)**, pp. 59-67, London, UK, 2000.

[3] B. Goethals, "Memory Issues in Frequent Itemset Mining", **In Proc. of the 2004 ACM Symposium on applied computing**, pp. 530-534, Nicosia, Cyprus, 2004.

[4] C. Lucchese, S. Orlando, and R. Perego, "Fast and Memory Efficient Mining of Frequent Closed Itemsets", **IEEE Journal Transactions on Knowledge and Data Engineering**, Vol. 18, No. 1, pp. 21–36, January 2006.

[5] C. Lucchesse, S. Orlando, and R. Perego, "DCI-Closed: A Fast and Memory Efficient Algorithm to Mine Frequent Closed Itemsets", **In Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations**, Vol. 126 of CEUR Workshop Proceedings, Brighton, UK, 1 November, 2004.

[6] C. Wu, "Mining Top-K Frequent Closed Itemsets Is Not in APX", **In Proc. of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining**, pp. 435-439, 2006.

[7] C. Yu and Y. Chen, "Mining Sequential Patterns from Multidimensional Sequence Data", **IEEE Transactions on Knowledge and Data Engineering**, Vol. 17, No. 1, pp. 136-140, January 2005.

[8] G. Grahne and J. Zhu, "Efficiently Using Prefix-trees in Mining Frequent Itemsets", **In Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2003)**, CEUR Workshop Proceedings, pp. 103, 19 November 2003.

[9] H. Pinto, **"Multi-dimensional Sequential Pattern Mining"**, M. Sc. Thesis, Simon Fraser University, Canada, 2001.

[10] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal, "Multi-dimensional Sequential Pattern Mining", **In Proc. of ACM CIKM Information and Knowledge Management (ACM'01)**, pp. 81-88, Atlanto, Georgia, USA, 2001.

[11] J. Han, J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M-C Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth", **In Proc. of International Conference on Data Engineering (ICDE'01)**, pp. 215-224, Heidelberg, Germany, 2001.

[12] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation", **In Proc. of 2000 ACM SIGMOD International conference Management of Data**, pp. 1-12, May 2000.

[13] J. Han, J. Wang, Y. Lu and P. Tzvetkov, "Mining Top-k Frequent Closed Patterns without Minimum Support", **In Proc. of the 2002 IEEE International Conference on Data Mining**, pp. 211-218, Washington, DC, USA, 2002.

[14] J. Pei, J. Han, and R. Mao, "Closet: An Efficient Algorithm for Mining Frequent Closed Itemsets", **In Proc. of the ACM-SIGMOD International Workshop on Data Mining and Knowledge Discovery (DMKD 2000)**, Dallas, Texas, USA, pp. 21–30, 2000.

[15] J. Wang, and J. Han, "BIDE: Efficient Mining of Frequent Closed Sequences", **In Proc. of the 20th International Conference of Data Engineering**, Boston, pp. 79-90, USA, 2004.

[16] J. Wang, J. Han, Y. Lu and P. Tzvetkov, "TFP: An Efficient Algorithm for Mining Top-K Frequent Closed Itemsets", **Journal of IEEE Transaction on knowledge and data mining**, Vol. 17, No. 5, pp. 652-664, May 2005.

[17] J.Wang, J. Han, and J. Pei, "Closet+: Searching for the Best Strategies for Mining Frequent Closed Itemsets", **In Proc. of the 9th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003)**, Washington D.C., USA, pp 236–245, 24-27 August 2003.

[18] K. Beyer, and R. Ramakrishnan, "Bottom-up Computation of Sparse and Iceberg Cubes", **In Proc. of ACM-SIGMOD International Conference Management of Data (SIGMOD'99)**, pp. 359-370, Philadelphia, Pennsylvnia, USA, 1999.

[19] K. Chuang, J. Huang, and M. Chen, "Mining Top-K Frequent Patterns in the Presence of the Memory Constraint", **Journal of VLDB**, Vol. 17, pp 1321-1344, November 2007.

[20] L. Li, D. Zhai, and F. Jin, "GRG: An Efficient Method for Association Rules Mining on Frequent Closed Itemsets", **In Proc. of IEEE International Symposium on Intelligent Control (ISIC'03)**, Houston, Texas, pp. 854-859, 2003.

[21] M. El-Hajj, and O. R. Zaiane, "COFI Tree Mining: A New Approach to Pattern Growth with Reduced Candidacy Generation", **Workshop on Frequent Itemset Mining Implementation (FIMI03) in Conjunction with IEEE-ICMD**, 2003.

[22] M. J. Zaki and C. J. Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining", **In Proc. of the 2nd SIAM International Conference on Data Mining**, Arlington, Virginia, USA, pp. 34–43, April 2002.

[23] M. Plantevit, Y. W. Choong, A. Laurent, D. Laurent and M. Teisseire, "Mining Closed Multidimensional Sequential Patterns", **Document report**, 2006.

[24] M. Plantevit, Y. W. Choong, A. Laurent, D. Laurent and M. Teisseire, "M2SP: Mining Sequential Patterns among Several Dimensions", **Knowledge Discovery in Database PKDD**, Vol 3721, 2005, pp. 205-216.

[25] M. U. Arshad, M. N. Ayyaz, "Mining N-most Interesting Itemsets Using Support-Ordered Tries", **In Proc. of the IEEE International Conference on Computer Systems and Applications**, pp. 592-599, Dubai, 2006.

[26] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Efficient Mining of Association Rules Using Closed Itemset Lattices", **Journal of Information Systems**, Vol. 24, No.1, pp.25–46, 1999.

[27] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering Frequent Closed Itemsets for Association Rules", **In Proc. of 7th International Conference on Database Theory (ICDT 1999)**, LNCS, Vol. 1540, Springer, Verlag, Jerusalem, Israel, January 1999, pp. 398–416.

[28] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Mining Based for Associate Rules Using Galois Closed Sets", **In Research Report**, Computer Science Dept., Nice Sophia Antipolis University.

[29] P. Songram and V. Boonjing, "Closed Multidimensional Sequential Pattern Mining", **Internation journal of Knowledge Management Studies**, Vol. 2, No. 4, pp. 460-479, 2008.

[30] P. Songram, V. Boonjing and S. Intakosum, "Closed Multidimensional Sequential Pattern Mining", **In Proc. of the 3rd IEEE Int. Conf. on Information Technology : New Generations (ITNG'06)**, pp.512-517, Las Vegas, Nevada, USA, 10-12 April, 2006.

[31] P. Tzvetkov, X. Yan and J. Han, "TSP: Mining Top-K Closed Sequential Patterns", **In Proc. of the 3rd IEEE International Conference on Data Mining**, pp. 347-357, Nov. 2003.

[32] P. Tzvetkov, X. Yan and J. Han, "TSP: Mining Top-K Closed Sequential Patterns", **Journal of Knowledge and Information Systems**, Vol. 7, Issue 4, pp. 438-457, May 2005.

[33] R. Agrawal and R. Srikant, "Fast Algorithm for Mining Association Rules", **In Proc. of 20th Very Large Database**, pp. 487-499, 1997.

[34] R.Bhagwan and B. Lin, "Fast and Scalable Priority Queue Architecture for High-Speed Network Switches", **In Proc. of IEEE INFOCOM**, pp. 538-547, 2000.

[35] S. Ngan, T. Lam, R.C. Wong, and A. W. Fu, "Mining N-Most Interesting Itemsets without Support Threshold by COFI-tree", **Journal of Business Intelligence and Data Mining**, Vol. 1, No. 1. 2005.

[36] T.M. Quang, S. Oyanagi, and K. Yamazaki, "ExMiner: An Efficient Algorithm for Mining Top-K Frequent Patterns", **Advanced Data Mining and Applications,** Vol. 4093, pp. 436-447, Springer-verlag Berlin Heidelberg, 2006.

[37] T.M.,Quang, S. Oyanagi, and K. Yamazaki, "Mining the K-Most Interesting Frequent Patterns Sequentially", **Intelligent Data Engineering and Automated Learninig (IDEAL 2006)**, pp. 620-628, Berlin Heidelberg , 2006.

[38] T. Uno, T. Asai, Y. Uchida, and H. Arimura, "An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases", **In Proc. of the 7th International Conference on Discovery Science**, Padova, Italy, pp. 16–31, 2-5 October 2004.

[39] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets", **In Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2004)**, Volume126 of CEUR Workshop Proceedings, Brighton, UK, 1 November 2004.

[40] V. Boonjing and P. Songram, "Efficient Algorithms for Mining Closed Multidimensional Sequential Patterns", **In Proc. of the 4th Int. Conf. on Fuzzy Systems and Knowledge Discovery (FSKD'07)**, pp. 749-753, Haikou, China, 24-27August , 2007

[41] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining Closed Sequential Patterns in Large Databases", **In Proc. of the 3rd SIAM International Conference on Data Mining (SDM'03)**, pp. 166-177, 2003.

[42] Y. Hirate , E. Iwahashi, and H.Yaman, "TF2P-growth: An Efficient Algorithm for Mining Frequent patterns without any thresholds", **In Proc. of IEEE International Conference on Data Mining**, 2004.

[43] Y-L. Cheng and A. Fu, "An FP-tree approach for mining N-most interesting itemsets", **In Proc. of the SPIE Conference on Data Mining**, pp. 460-471, 2002.

[44] Y-L. Cheng and A. Fu, "Mining Frequent Itemsets without Support Threshold: With and Without Item Constraints", **Journal of IEEE Transactions on Knowledge and Data Engineering**, Vol. 16, No. 9, pp. 1052-1069, Setember 2004.

APPENDIX

APPENDIX A

Publications

1. P. Songram and V. Boonjing, "N-Most Interesting Closed Itemset Mining", In Proc. of the 3rd Int. Conf. on Convergence and Hybrid Information Technology, pp. 619-624, 11-13 November, Korea, 2008.
2. P. Songram and V. Boonjing, "Mining Top-K Closed Itemsets Using Best-First Search", In Proc. of the 8rd IEEE Int. Conf. on Computer and Information Technology (CIT'08), pp. 77-82, Syney, Australia, 8-11 July, 2008.
3. P. Songram and V. Boonjing, "Closed multidimensional sequential pattern mining", Internation Journal of Knowledge Management Studies, Vol. 2, No. 4, pp. 460-479, 2008.
4. V. Boonjing and P. Songram, "Efficient Algorithms for Mining Closed Multidimensional Sequential Patterns",In Proc. of the 4th Int. Conf. on Fuzzy Systems and Knowledge Discovery (FSKD'07), pp. 749-753, Haikou, China, 24-27August, 2007.

Third 2008 International Conference on Convergence and Hybrid Information Technology

N-Most Interesting Closed Itemset Mining

Panida Songram[1,2] Veera Boonjing[1]
[1]*Software Systems Engineering Laboratory,*
Department of Mathematics and Computer Science, Faculty of Science,
King Mongkut's Institute of Technology Ladkrabang,
Bangkok, Thailand, kbveera@kmitl.ac.th
[2]*Department of Computer Science, Faculty of Informatics,*
Mahasarakham University, Mahasarakham, Thailand, panida.s@msu.ac.th

Abstract

In this paper, we propose an efficient algorithm, NCLOSED, for mining the N k-closed itemsets with the highest supports for 1 up to a certain k_{max} value. The algorithm adopts best-first search strategy to generate closed itemsets with highest remaining supports. It does not keep closed itemsets mined in main memory to ensure that they are really closed. This is because this algorithm can directly generate closed itemsets. Moreover, duplicated closed itemsets are detected and discarded from this algorithm.

1. Introduction

Closed itemset mining and closed sequence mining were proposed to eliminate redundant patterns generated in itemset mining and sequence mining, respectively. They only mine the patterns having no superset with the same support. They can reduce the number of patterns generated without information loss. Both these minings require a minimum support threshold as a filter. A too small minimum support threshold could give a large number of resulting patterns, whereas a too big one may often generate no answer. Moreover, an appropriate minimum support threshold is hard for users to set, because they need to be familiar with both the mining query and the task-specific data. To avoid these problems, Cheung et al.[13, 14] proposed *N*-most interesting itemsets mining. This task mines only the *N* *k*- itemsets with highest support for 1 up to a certain k_{max}, where k_{max} is the upper bound of the length of itemsets, and *N* is the desired number of *k*-itemsets. Three algorithms were proposed for this mining: LOOPBACK, BOLB, and BOMO. All three algorithms are adapted from the FP-tree approach. BOMO has two phases. First, it builds the complete FP-tree with all items in the database to find minimum support threshold of each *k*-itemset. Then it mines itemsets. During the mining process the support threshold of all itemsets is increased by considering the minimum value among the support of the N^{th} most frequent k-itemset discovered. It is used to prune unpromising itemsets. LOOPBACK builds the FP-tree and initializes the support threshold to be the support of the N^{th} sorted largest 1-frequent. If the number of any *k*-itemsets is less than *N*, the tree will be rebuilt to find the smaller support in order to mine more itemsets in the mining phase. BOLB is a hybrid approach of BOMO and LOOPBACK. Like BOMO, it builds the complete FP-tree only once. The mining process is applied from the technique of LOOPBACK. Wang et al.[5, 6] proposed mining top-*k* frequent closed itemsets of length no less than *min_l*, where *k* is the desired number of frequent closed itemsets to be mined, and *min_l* is the minimal length of closed itemsets. TFP starts with minimum support threshold at 0. It constructs an FP-tree to raise the threshold and uses the threshold to prune the tree. It may take a long time to construct the FP-tree to find the final threshold and to prune the tree if the database contains many transactions and long patterns. Moreover, TFP has to maintain candidates to ensure that they are really closed. Top-*k* closed itemset mining was extended to mine top-*k* closed sequence in [9, 10]. Pietracaprina et al.[1] proposed TopKMiner to mine top-*k* closed itemsets without considering the minimal length of them. From this mining, it can allow a user to dynamically raise the value *k* with no need to restart the computation. TopKMiner mines top-*k* closed itemsets by combining the LCM algorithm [11, 12] and priority queue to avoid closed checking. In addition, it adopts best first search to generate closed itemsets with highest support first. This idea is supported by C. Wu [4]. He proved that a heuristic algorithm is preferred over an exact algorithm to solve top-*k* closed itemset mining

N-most interesting itemsets mining may produce redundant itemsets. Some resulting itemsets are absorbed by other resulting itemsets. Although top-*k* closed itemset mining generates only non-redundant itemsets, it may not produce itemsets with high support if a large minimal

978-0-7695-3407-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ICCIT.2008.66

length is given. Actually, they are very interesting ones. Whereas 1-itemsets with high supports may be only generated if minimum length is not specific in the miming. In this paper, we propose a new mining task to answer the problems, called *N*-most interesting closed itemset mining. This work mines the *N* *k*-closed itemsets with highest supports for 1 up to a certain k_{max}, where k_{max} is the upper bound of the length of closed itemsets, and *N* is the desired number of *k*-itemsets. This work adopts a best-first search strategy to mine a closed itemset with highest support first, then mines closed itemsets with the highest remaining supports. Consequently, it is efficient in pruning unpromising itemsets.

The remainder of the paper is organized as follows. In section 2, we mention background and the basic definitions. The proposed method is given in section 3. Section 4 gives conclusion and future work.

2. Basic definitions

Let $D = \{t_1, t_2, ..., t_m\}$ be set of all transaction ids in the database, and $I = \{i_1, i_2, ..., i_n\}$ be set of all items appearing in the database. A set of transaction ids $T = \{t_1, t_2, ..., t_h\}$ is a non-empty subset of *D*, where $h \le m$. An itemset (set of items) $X = \{i_1, i_2, ..., i_l\}$ is a non-empty subset of *I*, where $l \le n$. Functions *f* and *g* are defined as follows.

$$f(T) = \{i \in I \mid \forall t \in T, i \in t\}$$

$$g(X) = \{t \in D \mid \forall i \in X, i \in t\}$$

The function *f* returns an itemset included in all the transaction ids belonging to *T*, and the function *g* returns the set of transaction ids supporting a given itemset *I*.

Definition 1 An itemset *X* is closed if and only if $\varsigma(X) = f(g(X)) = fog(X) = X$, where the composite function $\varsigma = fog$ is called a Galois operator or closure operator.

Example 1 Consider Table 1, the characters *a-h* represent items in database.

Table 1. An example database

Tid	Items
1	*a b c d e f g h*
2	*a b c e f g*
3	*c d f g h*
4	*b c d f g*
5	*a c f*
6	*c f*
7	*a b c f g h*
8	*a b c h*
9	*d e h*

Considering itemset *ca*, the closure of itemset *ca* $\varsigma(\{ca\}) = f(g(\{ca\})) = \{1, 2, 5, 7, 8\} = \{ca\}$. Thus, *ca* is closed.

Definition 2 An equivalence class is a set of itemsets with the same closures.

Definition 3 A generator is a minimal itemset of an equivalence class.

Example 2 Itemsets *g*, *fg*, *cg* and *cfg* belong to the same equivalence class, because they have the same closure, *cfg*. The minimal itemset of this class is *g*. Therefore, the generator of this class is *g*.

Definition 3 An itemset $X = \{i_1, i_2, ..., i_k\}$ is a non-empty subset of *I*, where *k* is number of items or length of the itemset, denoted as *k-itemset*.

Definition 4 The support of itemset *X* is the number of transaction ids containing *X*, $|g(X)|$, denoted as *supp(X)*.

Definition 5 An itemset *X* is a frequent itemset if its support is not less than a minimum support.

Definition 6 A frequent itemset *X* is called a closed itemset if there is no proper superset with the same support.

Definition 7 A closed itemset *X* is an *N*-most interesting *k*-closed itemset, if its support is not lower than the support of the N^{th} closed itemset that is sorted by descending support values.

Definition 8 The *N*-most interesting closed itemsets is union of the *N*-most interesting *k*-closed itemsets for $1 \le k \le k_{max}$, where k_{max} is the upper bound of the desired length of itemset.

Example 3 Suppose $N = 3$ and $k_{max} = 3$, the set of 3 1-closed itemsets found is {*c:8, h:5, d:4*}, the set of 3 2-closed itemset found is {*cf:7, ca:5, cb:5*}, and the set of 3 3-closed itemset found is {*cfg:5, cfa:4, cab:4*} Therefore, the number of resulting 3-most interesting closed itemsets is {*c:8, h:5, d:4, cf:7, ca:5, cb:5, cfg:5, cfa:4, cab:4*}.

3. The NCLOSED algorithm

We now present NCLOSED, a fast and memory efficient algorithm for mining *N*-most interesting closed itemsets. We will first show how the algorithm can generate closed itemsets without keeping closed itemset candidates in section 3.1. The pseudo codes are show in Figure 3 and 4. We then describe how the algorithm can find closed itemsets with high supports in section 3.2. Increasing minimum support thresholds and stopping the mining process are explained in section 3.3. The pseudo codes of section 3.2 and 3.3 are shown in Figure 2. The pseudo closed of NCLOSED algorithm is shown in Figure 1. In addition, we will give an example in section 3.4. Complexity as well as correctness will be analyzed in section 3.5.

Algorithm: NCLOSED (D, N, k_{max})
Input: A database D, the upper bound of the length k_{max} and the desired number of k-itemsets N.
Output: N-most interesting closed itemsets *NCI*.
Method:
1. Scan database D and find set of transaction ids in each item
2. $s[1] = s[2] = \ldots = s[k_{max}] = 0$
3. $c[1] = c[2] = \ldots = c[k_{max}] = 0$
4. NCI = N-mine($\emptyset$, $\emptyset$, 0, s, c)

Figure 1. NCLOSED algorithm

Subroutine: N-mine (*NCI*, X, *limit*, s, c)
Input: N-most interesting closed itemset *NCI*, an itemset X, the support limitation *limit*, array s for storing the final threshold of each length, and array c for counting number of closed itemsets of each length.
Output: N-most interesting closed itemsets *NCI*.
Method:
1. if $supp(X) < \forall s[k]$ where $1 \le k \le k_{max}$
2. then return *NCI*
3. if X.len = k_{max} then return 0
4. $C = \{X \cup i_1, X \cup i_2, \ldots, X \cup i_n\}$; $i \in$ *pre-items*(X)
5. if C is empty then return 0
6. if *number*(C) = 1 then
7. $best = X \cup i_1$, $alt = \alpha$
8. else
9. sort all elements *in* C by their supports in descending
10. order
11. $best$ = the first element in C
12. alt = the second element in C
13. if $supp(best) < limit$ then return $supp(best)$
14. if !Generator_check($best$) then return 0
15. Y = Closure_calculate($best$)
16. $best = Y$
17. if $supp(Y) \ge s[Y.len]$ then
18. $NCI = NCI \cup Y$
19. $c[Y.len] = c[Y.len] + 1$
20. if $c[Y.len] == N$ then
21. $s[Y.len] = supp(Y)$
22. return NCLOSED (*NCI*, $best$, $max(limit, supp(alt))$, s, c)

Figure 2. N-mine subroutine

Subroutine: Generator_check(X)
Input: An itemset X
Output: If X is a non-duplicate generator, 1 is returned. Otherwise, 0 is returned.
Method:
1. while($supp(X) = supp(b_i)$) ; $b_i \in$ *post-items*(X)
2. if($g(X) \subseteq g(b_i)$)
3. return 0
4. return 1

Figure 3. Generator checking subroutine

Subroutine: Closure_calculate(X)
Input: A itemset X
Output: A Closure of X, C_X
Method:
1. $C_X = X$
2. while($supp(X) = supp(f_j)$) ; $f_j \in$ *pre-items*(X)
3. if($g(X) \subseteq g(f_j)$)
4. $C_X = C_X \cup f_j$
5. return C_X

Figure 4. Closure calculation subroutine

3.1 Closed itemset generation

The NCLOSED algorithm effectively generates only closed itemsets without keeping candidates by computing closure of generator based on lemma 1 [2, 3]. Prior to the determination, the duplicate generators are detected to avoid generating existing closed itemsets according to lemma 3.

Lemma 1 Given an itemset X and an item $i \in I$, $g(X) \subseteq g(\{i\})$ if and only if $i \in \varsigma(X)$.

Example 4 The transaction ids containing itemset fg is $g(\{fg\}) = \{1, 2, 3, 4, 7\}$ and the transaction ids containing item c is $g(\{c\}) = \{1, 2, 3, 4, 5, 6, 7, 8\}$. We can see that $g(\{fg\}) \subseteq g(\{c\})$. Therefore, $c \in \varsigma(\{fg\})$.

With lemma 1, the closure of each itemset can be calculated by checking the transaction ids. Since it has to check a current itemset with all items, all items are sorted by their supports in descending order. Therefore, only pre-items of the current itemset are considered to do the checking with the current itemset according to lemma 2. This is because that the supports of the pre-items are not less than the support of the current itemset.

Definition 9 Let $X = Y \cup \{i\}$ be a generator, where Y is a closed itemset, $i \in I$ and $i \notin Y$. A set of pre-items of generator X is defined as follows.

$$pre-items(X) = \{j \mid j \in I, j \notin X, and\ j \prec i\}$$

Observation: $j \prec i$ means that j appears before i in sorted items.

Lemma 2 Suppose X is a generator, If $g(X) \subseteq g(\{j\})$, then $\varsigma(X) = X \cup \{j\}$, where $j \in pre-items(X)$.

Proof Since $g(X) \subseteq g(\{j\})$, $g(X \cup \{j\}) = g(X)$.
This shows that $f(g(X \cup \{j\})) = f(g(X))$
, or $\varsigma(X \cup \{j\}) = \varsigma(X)$. We have $j \in \varsigma(X)$.
Therefore, $\varsigma(X) = X \cup \{j\}$. □

The closure of each itemset is produced by including pre-items whose transaction ids contain its transaction ids. However, the same closure may be calculated twice from two different generators. To avoid this problem, the

generator has to be checked for duplications before calculating its closure according lemma 3.

Definition 10 Let $X = Y \cup \{i\}$ be a generator, where Y is a closed itemset, $i \in I$ and $i \notin Y$, is said to be a non-duplicate generator if and only if $\varsigma(X) = X$.

Definition 11 Let $X = Y \cup \{i\}$ be a generator, where Y is a closed itemset, $i \in I$ and $i \notin Y$. A set of post-items of generator X is defined as follows.

$$post-items(X) = \{ j \mid j \in I, j \notin X, and\ i \prec j \}$$

Lemma 3 Let $X = Y \cup \{i\}$ be a generator, where Y is a closed itemset, $i \in I$ and $i \notin Y$. If $\exists j \in post-items(X)$ such that $g(X) \subseteq g(\{j\})$, then X is a duplicate generator.

Proof If $g(X) \subseteq g(\{j\})$, then $j \in \varsigma(X)$ according to lemma 1. Since $j \in post-items(X)$, $j \notin X$ according to definition 11. $j \in \varsigma(X)$ but $j \notin X$. So $\varsigma(X) \neq X$. Based on definition 10, X is a duplicate generator. □

Lemma 3 shows that duplicate generators can be detected by checking transaction ids with post-items. Since items are sorted by their supports in descending order, some post-items do not need duplicate checking. If the support of any post-items is less than the support of the current itemset, the transaction ids of the post-item cannot include the transaction ids of the current itemset. Therefore, post-items with lower support of the current itemset are unnecessary to do the checking.

3.2 Finding closed itemsets with highest supports

This algorithm adopts best-first search to find closed itemsets with high supports first. During the mining process, the support of the generator is compared with the support limitation. The generator with higher support is computed the closure first because it can lead to the closed itemset with high support according to lemma 4 and 5.

Lemma 4 The support of a generator is equal to the support of its closure.

Proof Suppose X is a generator. We know that $\varsigma(X) = f(g(X))$, then $g(\varsigma(X)) = g(f(g(X)))$. But $g(f(g(X))) = \varsigma'(g(X)) = g(X)$, then $g(\varsigma(X)) = g(X)$

So $|g(\varsigma(X))| = |g(X)|$.

Therefore, $supp(\varsigma(X)) = supp(X)$. □

Lemma 5 For generator X and Y, if $supp(X) > supp(Y)$, then $supp(\varsigma(X)) > supp(\varsigma(Y))$.

Proof Since $supp(X) > supp(Y)$, we know that $|g(f(g(X)))| > |g(f(g(Y)))|$. This shows that $|g(\varsigma(X))| > |g(\varsigma(Y))|$.

Therefore, $supp(\varsigma(X)) > supp(\varsigma(Y))$. □

3.3 Increasing minimum support thresholds and stopping the mining process

This algorithm uses a simple structure, an array, to increase the minimum support thresholds of each length. An index in an array corresponds to the length of closed itemsets. Two arrays are used in this algorithm. The first one is used for counting the number of closed itemsets of each length. The second one is exploited for storing the final threshold of each length. Firstly, all elements in both arrays are initialized to zero. If a closed itemset is found, the element corresponding length of the itemset is increased by 1. After the element is increased till N, it is replaced by the support of the N^{th} closed itemset. The thresholds in the array are used to consider the remaining answer. Moreover, they are exploited to stop the mining process. If an itemset having lower support of all the thresholds is found, the mining process will be stopped. In addition, we will not extend closed itemsets having length k_{max} because it will lead to itemsets having length more than k_{max} that are not answers.

3.4 An example

Suppose we want to mine 3-most interesting 3-closed itemsets from the database as shown in Table 1. The database is scanned at once to find items with their transaction ids as shown in Table 2.

Table 2. Transaction ids of items

Item	Set of tids
c	1, 2, 3, 4, 5, 6, 7, 8
f	1, 2, 3, 4, 5, 6, 7
a	1, 2, 5, 7, 8
b	1, 2, 4, 7, 8
g	1, 2, 3, 4, 7
h	1, 3, 7, 8, 9
d	1, 3, 4, 9
e	1, 2, 9

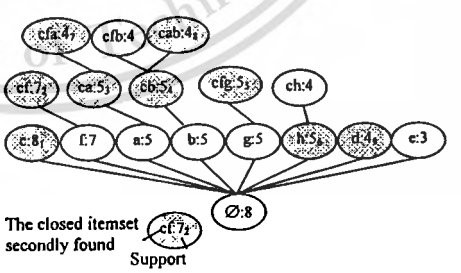


Figure 5. 3-most interesting closed itemsets

First, we initialize $NCI = \emptyset$, $X = \emptyset$, *limit* = 0, all element in array s and c = 0. Then all items are sorted by their supports in descending order. *c:8, f:7, a:5, b:5, g:5, h:5, d:4, e:3*.

Item *c* is firstly considered to be closed itemset. Item *f* is considered as an alternative item because its support is the second highest support. Item *c* is a closed item, so the element in array c is indexed by length of *c*, c[1], is increased by 1. Since item *c* cannot be extended because it does not have any pre-item. The support of *c* is set to 0 because it is no longer considered. Therefore, item *f* is the next to be considered. The lower support itemset, item *a*, is chosen as an alternative item. The support of *f* is not lower than the support limitation (*limit*). Thus, *f* is considered whether it is a generator. The transaction ids of *f* are not included in any transaction ids of its post-items, *a*, *b*, *g*, *h*, *d*, and *e*. Thus, it is a non-duplicate generator. Then, its closure is calculated by including its pre-item which contains transaction ids of *f*. Only pre-item *c* is added to *f* as closed itemset *cf:7*. Then, *f* is replaced by the closed itemset *cf* and c[2] is increased by 1. Since *cf* cannot be expanded to any itemset because there are no pre-items, the support of *cf* is set to 0. It is no longer considered.

The next consideration is item *a*. The alternative item is item *b*. The support of *a* is not less than the support limitation. Therefore, *a* is checked whether it is a non-duplicate generator and determined its closure. Item *a* is a non-duplicate generator and its closure is *ca:5*. Item *a* is replaced by *ca* and c[2] is increased by 1 as 2. The support limitation is reset to maximum value between the previous support limitation, 0, and the support of the alternative item *b*, 5. So the support limitation is 5. The closed itemset *ca* is extended with pre-item *f* as *cfa*. Since there is only one the extended itemset, the alternative itemset is set to α. We assign that the support of α is 0. Since the support of *cfa* is less than the support limitation, the support of *ca* is replaced by the support of *cfa*. The limitation support item, item *b*, is considered for finding the next closed itemset.

Now, we look back to the first level in order to consider item *b*. The alternative itemset is item *g* because *g* is the item with highest remaining support. The support of *b* is higher than the limitation support, 0. The item *b* is a non-duplicate generator. Its closure is *cb:5*. Then item *b* is replaced by the closure and c[2] is increased by 1 as 3. When c[2] is 3, s[2] is replaced by the support of *cb* that is 5. The support limitation is the support of *g* because its support is higher than the previous support limitation, 0. The closed itemset *cb* is extended with item *f* and *a* to be *cfb:4* and *cab:4*. The itemset *cfb* is considered as the best support, but its support is less then support limitation. The support of *cb* is then 4. Itemset *cb* is stopped considering the next closed itemset, and the support limitation item, *g*, is then considered to find closed itemsets. Item *g* leads to closed itemset, *cfg*:5. c[3] is increased by 1 as 1. The closed itemset *cfg* is not expanded because it will contribute itemsets having length more than 3. As soon as the N^{th} *k*-closed itemsets of each length are found, The thresholds in array are replaced by the support of them. NCLOSED recursively considers the remaining itemsets until an itemset having support lower than all the thresholds is found. The mining process will be stopped.

3.5 Complexity and Correctness

This section shows space complexity and time complexity of the proposed algorithm in Theorem 1 and 2, respectively. The correctness is shown in Theorem 3 and 4.

Theorem 1 The space complexity of NCLOSED is $O(bd)$; where b is the number of pre-items and d is the number of distinct items.

Proof Since this NCLOSED adopts recursive best-first search to mine N-most interesting closed itemsets. Therefore, its space complexity is $O(bd)$. □

Theorem 2 The worst case time complexity of NCLOSED is $O(b^{2d-1})$; where b is the number of pre-items and d is the number of distinct items

Proof. Since this NCLOSED adopts recursive best-first search to mine N-most interesting closed itemsets. The worst case time complexity of recursive best-first search is $O(b^{2d-1})$. Therefore, the time complexity of NCLOSED is $O(b^{2d-1})$. □

Theorem 3 An itemset having support lower than all support of the N^{th} k-closed itemset for $1 \le k \le k_{max}$ is not N-most interesting closed itemsets.

Proof Let X be an itemset, $s_k \in S$ is the support of the N^{th} k-closed itemset and $supp(X) < \forall s_k$ for $1 \le k \le k_{max}$. This method finds closed itemsets by calculating closure of generator. Based on lemma 4, $supp(\varsigma(X)) = supp(X)$, then $supp(\varsigma(X)) < \forall s_k$. Therefore, $\varsigma(X)$ is not any N-most interesting k-closed itemset according to definition 7. We can conclude that $\varsigma(X)$ is not N-most interesting closed itemset. □

Theorem 4 A closed itemset certainly expands any itemset whose support is less than its support.

Proof Let Y be a closed itemset, the extended itemset $X = Y \cup \{i\}$. According lemma 1, $i \notin Y$, $g(\{i\}) \neq g(Y)$. Therefore, $g(Y \cup \{i\}) \subset g(Y)$ and $|g(Y \cup \{i\})| \lhd g(Y)|$.

Thus, $supp(X) < supp(Y)$. □

Theorem 3 and 4 show that NCLOSED can be stopped mining process as soon as an itemset having lower support of all N^{th} k-closed itemsets is found. The remaining itemsets are unnecessarily determined, because NCLOSED has already found all resulting itemsets.

4. Conclusion and future work

This paper is a study on *N*-most interesting closed itemset mining. We also propose a new efficient method, called NCLOSED, for mining *N*-most interesting closed itemsets. The method adopts a best-first search technique to find the closed itemsets with high supports. The *N*-most interesting closed itemsets is found in order of descending supports. After the N^{th} closed itemset of each length is found, its supports is used as the final threshold for determining the remaining *N*-most interesting closed itemsets of such length. Moreover, all final thresholds of all length are used to stop the mining process. For traveling, each traversed itemset is checked whether it is a non-duplication generator or not. If it is a non-duplicate generator, its closure is calculated to find closed itemset. From this algorithm, its implications are: 1) it efficiently finds *N*-most interesting closed itemsets, 2) it does not need to find the final thresholds before mining (the all final thresholds are found when all the N^{th} *k*-closed itemsets found), 3) it is an efficient pruning of unpromising itemsets and stopping rapidly, and 4) it does not need to maintain closed itemsets mined in memory (it does not require closed checking).

However, calculating the closure of generator may give closed itemsets that are not the expected answers, but they are required to reach the remaining *N*-most interesting closed itemsets.

Acknowledgment: The authors would like to thank Richard Booth for useful comments and proof-reading the English.

5. References

[1] A. Pietracaprina and F.Vandin, "Efficient Incremental Mining of Top-k Frequent Closed Itemsets", In Pro of 10th International Conference on Discovery Science, pp. 275-280, Sendai, Japan, 1-4 October, 2007.

[2] C. Lucchese, S. Orlando, and R. Perego, "Fast and Memory Efficient Mining of Frequent Closed Itemsets", *In IEEE Journal Transactions on Knowledge and Data Engineering*, 18 (1):21–36, January 2006.

[3] C. Lucchesse, S. Orlando, and R. Perego, "DCI-Closed: A Fast and Memory Efficient Algorithm to Mine Frequent Closed Itemsets", In Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Vol. 126 of CEUR Workshop Proceedings, Brighton, UK, 1 November, 2004.

[4] C. Wu, "Mining Top-K Frequent Closed Itemsets Is Not in APX", In Proc. of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2006, pp. 435-439.

[5] J. Wang, J. Han, Y. Lu and P. Tzvetkov, "TFP: An Efficient Algorithm for Mining Top-K Frequent Closed Itemsets", *Journal of IEEE Transaction on knowledge and data mining*, Vol. 17, No. 5, May 2005, pp. 652-664.

[6] J. Han, J. Wang, Y. Lu and P. Tzvetkov, "Mining Top-k Frequent Closed Patterns without Minimum Support", In Proc. of the 2002 IEEE International Conference on Data Mining, Washington, DC, USA, 2002, pp. 211-218.

[7] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Efficient Mining of Association Rules Using Closed Itemset Lattices", *Journal of Information Systems*, 24(1): 1999, pp.25–46.

[8] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering Frequent Closed Itemsets for Association Rules", In Proc. of 7th International Conference on Database Theory, LNCS, Vol. 1540, Springer, Verlag, Jerusalem, Israel, January 1999, pp. 398–416.

[9] P. Tzvetkov, X. Yan and J. Han, "TSP: Mining Top-K Closed Sequential Patterns", In Proc. of the 3rd IEEE International Conference on Data Mining, Nov. 2003, pp. 347-357.

[10] P. Tzvetkov, X. Yan and J. Han, "TSP: Mining Top-K Closed Sequential Patterns", *Journal of Knowledge and Information Systems*, Vol. 7, Issue 4, May 2005, pp. 438-457.

[11] T. Uno, T. Asai, Y. Uchida, and H. Arimura, "An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases", In Proc. of the 7th International Conference on Discovery Science, Padova, Italy, 2-5 October 2004, pp. 16–31.

[12] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets", In Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Vol. 126 of CEUR Workshop Proceedings, Brighton, UK, 1 November 2004.

[13] Y-L. Cheng and A. Fu, "An FP-tree approach for mining N-most interesting itemsets", In Proc. of the SPIE Conference on Data Mining, pp. 460-471, 2002.

[14] Y-L. Cheng and A. Fu, "Mining Frequent Itemsets without Support Threshold: With and Without Item Constraints", *Journal of IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 9, pp. 1052-1069, Setember 2004.

Mining Top-K Closed Itemsets Using Best-First Search

Panida Songram[1,2] Veera Boonjing[1]
[1]*Software Systems Engineering Laboratory,*
Department of Mathematics and Computer Science, Faculty of Science,
King Mongkut's Institute of Technology Ladkrabang,
Bangkok, Thailand, kbveera@kmitl.ac.th
[2]*Department of Computer Science, Faculty of Informatics,*
Mahasarakham University, Mahasarakham, Thailand, panida.s@msu.ac.th

Abstract

In this paper, we proposed an efficient algorithm, TOPK_CLOSED, for mining top-k closed itemsets. This algorithm mines top-k closed itemsets using best-first search. The closed itemsets with the highest supports are firstly found from this algorithm. Consequently, the method quickly finds top-k closed itemsets, which leads to an efficient pruning unnecessary itemsets and stop mining rapidly. In addition, this algorithm can generate closed itemsets without keeping candidates in main memory.

1. Introduction

Closed itemset mining and closed sequence mining were proposed to eliminate redundant patterns generated in itemset mining and sequence mining, respectively. They only mine the patterns having no superset with the same support. They can reduce the number of patterns generated without information loss. Both of minings require a minimum support threshold as a filter. A too small minimum support threshold could give a large number of resulting patterns, whereas a too big one may often generate no answer. Moreover, an appropriate minimum support threshold is hard for users to set, because they need to be familiar with both mining query and the task-specific data. To avoid these problems, Cheung et al.[13, 14] proposed to mine only N k- itemsets with highest support for k up to a certain k_{max}, where k_{max} is the upper bound of the size of itemsets, and N is the desired number of k-itemsets. Three algorithms were proposed for this mining: LOOPBACK, BOLB, and BOMO. Wang et al.[5, 6] proposed mining top-k frequent closed itemsets of length no less than *min_l*, where k is the desired number of frequent closed itemsets to be mined, and *min_l* is the minimal length of each itemset. TFP is started minimum support threshold at 0. It constructs FP-tree to raise the threshold and uses the threshold to prune the tree. It may take a long time to construct FP-tree, to find the final threshold and to prune the tree if the database contains many transactions and long patterns. Moreover, TFP has to maintain candidates to ensure that they are really closed. Top-k closed itemset mining was extended to mine top-k closed sequence in [9, 10]. Some ideas in top-k closed itemset mining were used in top-k closed sequence mining. However, top-k closed sequence mining has to test subsequence required order matching which is more difficult than subset testing. Pietracaprina et al.[1] proposed TopKMiner to mine top-k closed itemsets without considering the minimum length of the itemsets. From this mining, it can allow a user to dynamically raise the value k with no need to restart the computation. TopKMiner mines top-k closed itemsets by combining LCM algorithm [11, 12] and priority queue to avoid closed checking. In addition, it adopts best first search to generate closed itemsets with highest support first. This idea is supported by C. Wu [4]. He proved that heuristic algorithm is preferred to solve top-k closed itemset mining than exact algorithm. TopKMiner is shown that it is better performance than TFP when mining top-k closed itemsets without minimum length constraint. However, the algorithm may produce only 1-itemsets with high supports if there are many distinct items found in many transactions.

In this paper, we propose best-first search strategy to mine top-k closed itemsets of length no less than *min_l*. The algorithm can mine the closed itemset with highest support first, then mine closed itemsets with the highest remaining supports. As soon as the itemset having lower support k^{th} closed itemset is found, the mining will be stopped. This is because we have

978-1-4244-2358-3/08/$20.00 © 2008 IEEE

already found the top-k closed itemsets. Consequently, it is efficient in pruning unpromising itemsets.

The remainder of the paper is organized as follows. In section 2, we give background and basic definitions. The proposed method is described in section 3. Section 4 concludes the paper.

2. Background and basic definitions

2.1. Background

Let $D = \{t_1, t_2, ..., t_m\}$ be set of all transaction ids in database, and $I = \{i_1, i_2, ..., i_n\}$ be set of all items appearing in database. A set of transaction ids $T = \{t_1, t_2, ..., t_h\}$ is a non-empty subset of D, where $h \le m$. An itemset (set of items) $X = \{i_1, i_2, ..., i_l\}$ is a non-empty subset of I, where $l \le n$. Functions f and g are defined as follows.

$$f(T) = \{i \in I \mid \forall t \in T, i \in t\}$$

$$g(X) = \{t \in D \mid \forall i \in X, i \in t\}$$

The function f returns an itemset included in all the transaction ids belonging to T, and the function g returns the set of transaction ids supporting a given itemset I.

Definition 1. An itemset X is closed if and only if $\varsigma(X) = f(g(X)) = fog(X) = X$, where the composite function $\varsigma = fog$ is called Galois operator or closure operator.

Example 1. Consider Table 1, the closure of itemset *ca* $\varsigma(\{ca\}) = f(g(\{ca\})) = \{1, 2, 5, 7, 8\} = \{ca\}$. Thus, *ca* is closed.

Table 1. An example database

Tid	Items
1	a b c d e f g h
2	a b c e f g
3	c d f g h
4	b c d f g
5	a c f
6	c f
7	a b c f g h
8	a b c h

Definition 2. An equivalence class is a set of itemsets with the same closures.

From definition 2, the maximal itemset in each class is closed itemset. Thus, algorithms such as, CLOSE [7], A-CLOSE [8], DCI_CLOSED [2, 3] and LCM [11, 12], try to generate closed itemsets by finding maximal itemset in each class.

Definition 3. A generator is a minimal itemset of equivalence class.

Example 2. Itemset *g*, *fg*, *cg* and *cfg* belong to the same equivalence class, because they have the same closure, *cfg*. The minimal itemset of this class is *g*. Therefore, the generator of this class is *g*.

CLOSE, A-CLOSE, DCI_CLOSED and LCM directly find closed itemsets by calculating closure of the generators. They do not keep closed itemset candidates to check whether the candidates are really closed itemsets. However, the redundant computation may occur; the same closed itemset may be calculated twice from two different generators. This problem occurs with CLOSE and A-CLOSE. DCI_CLOSED and LCM can avoid this problem. They can detect and discard duplicate generators to avoid the duplicate calculation of the same closed itemset, only one minimal element of each class is determined to find closed itemsets. Our proposed algorithm employs this idea to generate top-k closed itemsets.

2.2. Basic definitions

Definition 4. Length of itemset X is a number of items, denoted as *l-itemset*.

Definition 5. The support of itemset X is a number of transaction ids containing X, denoted as *supp(X)* or $|g(X)|$.

Definition 6. An itemset X is a frequent itemset if its support is not less than a minimum support threshold.

Definition 7. A frequent itemset X is a closed itemset if there is no proper superset with the same support.

Definition 8. A closed itemset X is a top-k closed itemset of minimal length *min_l* if there is no more than (k-1) closed itemset of length at least minimum length *min_l* whose support is higher than that of X.

Example 3. Suppose we want to find top-5 frequent closed itemsets of length at least 2. The set of first five closed itemsets found is {*cf:7, ca:5, cb:5, cfg:5, cfbg:4*}, but the number of resulting itemsets could contain more than five elements because there are more than one itemset of length at least *min_l* having the same support. Thus, The number of resulting itemsets is {*cf:7, ca:5, cb:5, cfg:5, cfbg:4, ch:4, cfa:4, cab:4*}.

3. The TOPK_CLOSED algorithm

This algorithm is a fast and memory efficient algorithm for mining top-k closed itemsets. It adopts best-first search strategy to quickly find top-k closed itemsets with highest remaining support. Moreover, it generates only closed itemsets without keeping

candidates by determining closure of generator based on lemma 1 [2, 3]. Prior to the determination, the duplicate generators are detected to avoid generating existing closed itemsets according to lemma 3.

Lemma 1. Given an itemset X and an item $i \in I$, $g(X) \subseteq g(\{i\})$ if and only if $i \in \varsigma(X)$.

Example 4. The transaction ids containing itemset fg is $g(\{fg\}) = \{1, 2, 3, 4, 7\}$ and the transaction ids containing item c is $g(\{c\}) = \{1, 2, 3, 4, 5, 7, 8\}$. We can see that $g(\{fg\}) \subseteq g(\{c\})$. Therefore, $c \in \varsigma(\{fg\})$.

With lemma 1, the closure of each itemset can be calculated by checking of transaction ids. Since it is suffering to do the checking a current itemset with all items, all items are sorted by their supports in descending order. Therefore, only pre-items of the current itemset are considered to do the checking with the current itemset according to lemma 2. This is because that the supports of the pre-items are not less than the support of the current itemset.

Definition 9. A generator $X = Y \cup \{i\}$, where Y is a closed itemset, $i \in I$ and $i \notin Y$. A set of pre-items of generator X is defined as follows.

$$pre-items(X) = \{j \mid j \in I, j \notin X, and\ j \prec i\}$$

Observation: $j \prec i$ means that j appears before i in sorted items.

Lemma 2. Suppose X is a generator, If $g(X) \subseteq g(\{j\})$, then $\varsigma(X) = X \cup \{j\}$, where $j \in pre-items(X)$.

Proof. Since $g(X) \subseteq g(\{j\})$, $g(X \cup \{j\}) = g(X)$. This shows that $f(g(X \cup \{j\})) = f(g(X))$, or $\varsigma(X \cup \{j\}) = \varsigma(X)$. We have $j \in \varsigma(X)$. Therefore, $\varsigma(X) = X \cup \{j\}$. □

The closure of each itemset is produced by including pre-items whose transaction ids contain its transaction ids. However, the same closure may be calculated twice from two different generators. To avoid this problem, the generator has to be checked duplication before calculating its closure according lemma 3.

Definition 10. A generator $X = Y \cup \{i\}$, where Y is a closed itemset, $i \in I$ and $i \notin Y$, is said to be a non-duplicate generator if and only if $\varsigma(X) = X$.

Definition 11. A generator $X = Y \cup \{i\}$, where Y is a closed itemset, $i \in I$ and $i \notin Y$. A set of post-items of generator X is defined as follows.

$$post-items(X) = \{j \mid j \in I, j \notin X, and\ i \prec j\}$$

Lemma 3. A generator $X = Y \cup \{i\}$, where Y is a closed itemset, $i \in I$ and $i \notin Y$. If $\exists j = post-items(X)$ such that $g(X) \subseteq g(\{j\})$, then X is a duplicate generator.

Proof. If $g(X) \subseteq g(\{j\})$, then $j \in \varsigma(X)$ according to lemma 1. Since $j \in post-items(X)$, $j \notin X$ according to definition 11. $j \in \varsigma(X)$ but $j \notin X$. So $\varsigma(X) \neq X$. Based on definition 10, X is a duplicate generator. □

Lemma 3 shows that duplicate generator can be detected by checking transaction ids with post-items. Since items are sorted by their supports in descending order, some post-items do not need duplicate checking. If support of any post-items is less than the support of the current itemset, the transaction ids of the post-item cannot include the transaction ids of the current itemset. Therefore, post-items with lower support of the current itemset are unnecessary to do the checking.

During mining process, the support of generator is compared with the support limitation. The generator with higher support is computed the closure first because it can lead to the closed itemset with high support according to lemma 4 and 5. Therefore, top-k closed itemsets can be found early. This leads to pruning unnecessary itemsets and stopping mining rapidly.

Lemma 4. The support of a generator is equal to the support of its closure.

Proof. Suppose X is a generator. We know that $\varsigma(X) = f(g(X))$, then $g(\varsigma(X)) = g(f(g(X)))$. But $g(f(g(X))) = \varsigma'(g(X)) = g(X)$, then $g(\varsigma(X)) = g(X)$ So $|g(\varsigma(X))| = |g(X)|$.

Therefore, $supp(\varsigma(X)) = supp(X)$. □

Lemma 5. For generator X and Y, if $supp(X) > supp(Y)$, then $supp(\varsigma(X)) > supp(\varsigma(Y))$.

Proof. Since $supp(X) > supp(Y)$, we know that $|g(f(g(X)))| > |g(f(g(Y)))|$. This shows that $|g(\varsigma(X))| > |g(\varsigma(Y))|$.

Therefore, $supp(\varsigma(X)) > supp(\varsigma(Y))$. □

The TOPK_CLOSED algorithm is as shown in Figure 1, 2 and 3. It mines top-k closed itemsets using the following steps. First, database is scanned at once to find 1-itemset with their transaction ids. A transaction id with length less than *min_l* is not considered in mining top-k closed itemsets, because it cannot contribute to a closed itemset of minimum length *min_l*. For example, if *min_l* is 4, the transaction 5 and 6 will not be considered to be top-k closed itemsets. The procedure TOPK_CLOSED is then called by passing as arguments: $TCI = \emptyset$, $X = \emptyset$, $limit = 0$, $suppK = 1$ and $m = 0$. The procedure expands all possible generators (line 2), and sort them by their

support in descending order (line 7). For the first level, the min_l^{th} 1-itemset is determined first (line 8), because the items before min_l^{th} 1-itemset cannot produce the closed itemsets of length at least minimum length *min_l*. In addition, the min_l^{th} 1-itemset tends to produce a closed itemset with highest support. Each current itemset (node) is checked that its support is lower than support limitation or not (line 12). If it has the lower support, the support of current itemset is replaced by the highest support of its extended itemset. Otherwise, the itemset is checked whether it is a non-duplicate generator (line 13) before calculating its closure (line 14). If it is the non-duplicate generator, the closure is calculated to find closed itemset. The closed itemset of length no less than *min_l* are included in the set of top-*k* closed itemsets(line 17). After a closed itemset is found, it will be used to expand itemsets in recursive. While mining process, the number of top-*k* closed itemset is counted (line 18) to reset the minimum support value (line 19-20). This mining is stopped when the itemset have lower support of k^{th} top-*k* closed itemset is found.

Algorithm: TOPK_CLOSED (*TCI, X, limit, suppK, m*)
Input: A database $\bar{D}$,
Output: top-*k* closed itemset *TCI*.
Method:
1. if *supp(X)*< *suppK* then return *TCI*
2. *children* = $\{X \cup i_1, X \cup i_2, X \cup i_3, \ldots, X \cup i_m\}$; $i \in$ *pre-items(X)* //expand *X*
3. if *children* is empty then return 0
4. if *number(children)* = 1 then //check number of children
5. *best* = *child*, *alt* = α
6. else
7. sort *children* by their supports in descending order
8. if first level
9. *best* = $child_{min\_l}$, *alt* = $child_{min\_l+1}$
10. else
11. *best* = $child_1$, *alt* = $child_2$
12. if *supp(best)* < *limit* then return *supp(best)*
13. if !Generator_check(*best*) then return 0
14. *C* = Closure_calculate(*best*)
15. *best* = *C*
16. if(*len(C)* ≥ *min_l*) //check length of closed itemset
17. *TCI* = *TCI* ∪ *C*
18. *m* = *m*+1
19. if(*m*==*k*)
20. *suppk* = the support of k^{th} top-*k* closed itemset
21. return TOPK_CLOSED (*TCI, best, max(limit,supp(alt)), suppK, m*)

Figure 1. TOPK_CLOSED algorithm

Subroutine: Generator_check(*X*)
Input: A itemset *X*
Output: If *X* is a non-duplicate generator, 1 is returned. Otherwise, 0 is returned.
Method:
1. while(*supp(X)*=*supp*(b_i)) ; $b_i \in$ *post-items(X)*
2. if($g(X) \subseteq g(b_i)$)
3. return 0
4. return 1

Figure 2. Generator checking subroutine

Subroutine: Closure_calculate(*X*)
Input: A itemset *X*
Output: A Closure of *X*, C_X
Method:
1. $C_X = X$
2. while(*supp(X)* = *supp*(j_j)) ; $f_j \in$ *pre-items(X)*
3. if($g(X) \subseteq g(j_j)$)
4. $C_X = C_X \cup f_j$
5. return C_X

Figure 3. Closure calculation subroutine

3.1. An example

Suppose we want to mine top-*5* closed itemsets with minimum length 2 from transactions shown in Table1. Database is scanned at once to find 1-itemsets with their transaction ids as shown in Table 2.

Table 2. Transaction ids of items

Item	Set of tids
c	1, 2, 3, 4, 5, 7, 8
f	1, 2, 3, 4, 5, 7
a	1, 2, 5, 7, 8
b	1, 2, 4, 7, 8
g	1, 2, 3, 4, 7
h	1, 3, 7, 8
d	1, 3, 4
e	1, 2

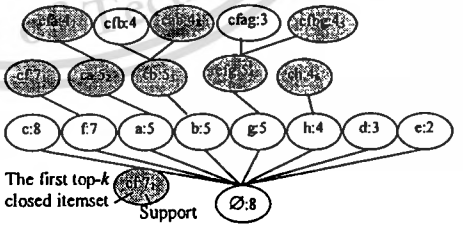


Figure 4. Top-*5* closed itemsets

First, we initial $TCI = \emptyset$, $X = \emptyset$, *limit* =0, *suppK* = 1 and $m = 0$. The all items are sorted by their supports in descending order. *c:8, f:7, a:5, b:5, g:4, h:4, d:3, e:2.*

Since item *c* cannot be extended to any closed itemset with length no less 2. Therefore, item *f* is firstly considered to find top-*k* closed itemsets. The lower support itemset, item *a*, is chosen as an alternative item. The support of *f* is not lower than the support limitation (*limit*). Thus, *f* is considered whether it is a generator. The transaction ids of *f* are not included in any transaction ids of its post-items, *a, b, g, h, d, and e*. Thus, it is a non-duplicate generator. Then, its closure is calculated by including its pre-item which contains transaction ids of *f*. Only pre-item *c* is added to *f* as closed itemset *cf:7*. The closed itemset is the top-*5* closed itemset with highest support. Then, *f* is replaced by the closed itemset *cf*. Since *cf* cannot be expanded to any itemset because there are no pre-items, the support of *cf* is set to 0. It is no longer considered.

The next consideration is item *a*. The alternative item is item *b*. The support of *a* is not less than the support limitation. Therefore, *a* is checked whether it is a non-duplicate generator and determined its closure. Item *a* is a non-duplicate generator and its closure is *ca:5*. Itemset *ca* is the second top-*5* closed itemset. Item *a* is replaced by *ca*. The support limitation is reset to maximum value between the previous support limitation, 0, and the support of the alternative item *b*, 5. So the support limitation is 5. The closed itemset *ca* is extended with pre-item *f* as *cfa*. Since there is only one the extended itemset, the alternative itemset is set to α. We assign that the support of α is 0. Since the support of *cfa* is less than the support limitation, the support of *ca* is replaced by the support of *cfa*. The limitation support item, item *b*, is considered for finding the next top-*k* closed itemset.

Now, we look back to the first level in order to consider item *b*. The alternative itemset is item *g* because *g* is the item with highest remaining support. The support of *b* is higher than the limitation support, 0. The item *b* is a non-duplicate generator. Its closure is *cb:5* which is the third top-*5* closed itemset. The item *b* is replaced by the closure. Now, the support limitation is the support of *g* because its support is higher than the previous support limitation, 0. The closed itemset *cb* is extended with item *f* and *a* to be *cfb:4* and *cab:4*. The itemset *cfb* is considered as the best support, but its support is less then support limitation. The support of *cb* is then 4. Itemset *cb* is stopped considering the next top-*5* closed itemset, and the support limitation item, *g*, is then considered to find top-*k* closed itemsets. Item *g* leads to fourth and fifth top-*5* closed itemset, *cfg:5* and *cfbg:4*.

As soon as the fifth top-*5* closed itemset has been found, its support is set to the final support threshold. TOPK_CLOSED recursively considers itemset *h*, *ca* and *cb* to find the remaining top-*5* closed itemsets having the same support of fifth top-*5* closed itemset. It obtains {*cf:7, ca:5, cb:5, cfg:5, cfbg:4, ch:4, cfa:4, cab:4*} as the top-*5* closed itemsets.

3.2. Complexity and Correctness

This section shows space complexity and time complexity of the proposed algorithm in Theorem 1 and 2, respectively. The correctness is shown in Theorem 3 and 4.

Theorem 1. The space complexity of TOPK_CLOSED is $O(bd)$; where b is the branching factor (number of children of each node) and d is the maximum search depth.

Proof. Since this TOPK_CLOSED adopts recursive best-first search to mine top-*k* closed itemsets. Therefore, its space complexity is $O(bd)$. □

Theorem 2. The worst case time complexity of TOPK_CLOSED is $O(b^{2d-1})$; where b is the branching factor, d is the maximum search depth.

Proof. Since this TOPK_CLOSED adopts recursive best-first search to mine top-*k* closed itemsets. The worst case time complexity of recursive best-first search is $O(b^{2d-1})$. Therefore, the time complexity of TOPK_CLOSED is $O(b^{2d-1})$. □

Theorem 3. An itemset having support lower than support of the k^{th} top-*k* closed itemset is not top-*k* closed itemsets.

Proof. Let X be an itemset with $supp(X) < suppK$, where *suppK* is the support of the k^{th} top-*k* closed itemset. Since this method finds top-*k* closed itemsets by calculating closure of generator. Based on lemma 4, $supp(\varsigma(X)) = supp(X)$, then $supp(\varsigma(X)) < suppK$. Therefore, $\varsigma(X)$ is not top-*k* closed itemset according to definition 8. □

Theorem 4. A closed itemset certainly expands any itemset whose support is less than its support.

Proof. Let Y be a closed itemset, the extended itemset $X = Y \cup \{i\}$. According lemma 1, $i \notin Y$, $g(\{i\}) \neq g(Y)$. Therefore, $g(Y \cup \{i\}) \subset g(Y)$ and $|g(Y \cup \{i\})| < g(Y)|$. Thus, $supp(X) < supp(Y)$. □

Theorem 3 and 4 show that TOPK_CLOSED can be stopped mining top-*k* closed itemsets as soon as an itemset having lower support of k^{th} top-*k* closed itemset is found. The remaining itemsets are unnecessarily determined to mine top-*k* closed itemsets, because TOPK_CLOSED has already found all resulting itemsets.

4. Conclusion and future work

This paper is a study on top-*k* closed itemset mining. We propose new efficient method, called TOPK_CLOSED. This method generates top-*k* closed itemsets using best-first search strategy. It can find the top-*k* closed itemset with highest support first. The top-*k* closed itemsets with the highest remaining supports are found later. In addition, each traversed itemset is checked whether it is a non-duplication generator. If it is a non-duplicate generator, its closure is calculated to find closed itemset. The implications of this algorithm are: 1) it fast finds top-*k* closed itemsets, 2) it does not need finding the final minimum support threshold before mining (the final support threshold is found when the k^{th} top-*k* closed itemset found), 3) it is an efficient pruning unpromising itemsets and stopping rapidly as soon as top-*k* closed itemsets are found, 4) it does not need to maintain the top-*k* closed itemsets mined in memory (it does not require closed checking), and 5) some itemsets are skipped length by calculating their closures.

However, calculating closure of generator may give closed itemsets of length less than minimum length. Although these closed itemsets are not top-*k* closed itemsets, they are required to reach closed itemsets of length no less than minimum length. If minimum length is high, top-*k* closed itemsets of length no less than minimum length may be deeply found.

Acknowledgment: The authors would like to thank Richard Booth for useful comments and proof-reading the English.

5. References

[1] A. Pietracaprina and F.Vandin, "Efficient Incremental Mining of Top-k Frequent Closed Itemsets", In Pro of 10th International Conference on Discovery Science, pp. 275-280, Sendai, Japan, 1-4 October, 2007.

[2] C. Lucchese, S. Orlando, and R. Perego, "Fast and Memory Efficient Mining of Frequent Closed Itemsets", *In IEEE Journal Transactions on Knowledge and Data Engineering*, 18 (1):21–36, January 2006.

[3] C. Lucchesse, S. Orlando, and R. Perego, "DCI-Closed: A Fast and Memory Efficient Algorithm to Mine Frequent Closed Itemsets", In Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Vol. 126 of CEUR Workshop Proceedings, Brighton, UK, 1 November, 2004.

[4] C. Wu, "Mining Top-K Frequent Closed Itemsets Is Not in APX", In Proc. of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2006, pp. 435-439.

[5] J. Wang, J. Han, Y. Lu and P. Tzvetkov, "TFP: An Efficient Algorithm for Mining Top-K Frequent Closed Itemsets", *Journal of IEEE Transaction on knowledge and data mining*, Vol. 17, No. 5, May 2005, pp. 652-664.

[6] J. Han, J. Wang, Y. Lu and P. Tzvetkov, "Mining Top-k Frequent Closed Patterns without Minimum Support", In Proc. of the 2002 IEEE International Conference on Data Mining, Washington, DC, USA, 2002, pp. 211-218.

[7] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Efficient Mining of Association Rules Using Closed Itemset Lattices", *Journal of Information Systems*, 24(1): 1999, pp.25–46.

[8] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering Frequent Closed Itemsets for Association Rules", In Proc. of 7th International Conference on Database Theory, LNCS, Vol. 1540, Springer, Verlag, Jerusalem, Israel, January 1999, pp. 398–416.

[9] P. Tzvetkov, X. Yan and J. Han, "TSP: Mining Top-K Closed Sequential Patterns", In Proc. of the 3rd IEEE International Conference on Data Mining, Nov. 2003, pp. 347-357.

[10] P. Tzvetkov, X. Yan and J. Han, "TSP: Mining Top-K Closed Sequential Patterns", *Journal of Knowledge and Information Systems*, Vol. 7, Issue 4, May 2005, pp. 438-457.

[11] T. Uno, T. Asai, Y. Uchida, and H. Arimura, "An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases", In Proc. of the 7th International Conference on Discovery Science, Padova, Italy, 2-5 October 2004, pp. 16–31.

[12] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets", In Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Vol. 126 of CEUR Workshop Proceedings, Brighton, UK, 1 November 2004.

[13] Y-L. Cheng and A. Fu, "An FP-tree approach for mining N-most interesting itemsets", In Proc. of the SPIE Conference on Data Mining, pp. 460-471, 2002.

[14] Y-L. Cheng and A. Fu, "Mining Frequent Itemsets without Support Threshold: With and Without Item Constraints", *Journal of IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 9, pp. 1052-1069, Setember 2004.

460 *Int. J. Knowledge Management Studies, Vol. 2, No. 4, 2008*

Closed multidimensional sequential pattern mining

Panida Songram* and Veera Boonjing

Software Systems Engineering Laboratory,
Department of Mathematics and Computer Science,
King Mongkut's Institute of Technology Ladkrabang,
Bangkok, Thailand
E-mail: s9062902@kmitl.ac.th
E-mail: kbveera@kmitl.ac.th
*Corresponding author

Abstract: We propose a new method, called closed multidimensional sequential pattern mining, for mining multidimensional sequential patterns. The new method is an integration of closed sequential pattern mining and closed itemset pattern mining. This integration is performed in two approaches: mining closed itemset patterns followed by mining closed sequential patterns and mining closed sequential patterns followed by mining closed itemset patterns. For the new method, it is shown that (1) the number of complete closed multidimensional sequential patterns is not larger than the number of complete multidimensional sequential patterns (2)the set of complete closed multidimensional sequential patterns covers the complete resulting set of multidimensional sequential patterns. In addition, mining using closed itemset pattern mining on multidimensional information would mine only multidimensional information associated with mined closed sequential patterns, and mining using closed sequential pattern mining on sequences would mine only sequences associated with mined closed itemset patterns. Moreover, we experimentally investigated using these two integration approaches to determine factors affecting the number of candidate patterns. The experiment results show that the first approach gives less number of patterns than the second on the datasets with few dimensions. For the data sets with many dimensions, the results are opposite.

Keywords: closed multidimensional pattern mining; closed itemset pattern mining; closed sequential pattern mining; data mining.

Reference to this paper should be made as follows: Songram, P. and Boonjing, V. (2008) 'Closed multidimensional sequential pattern mining', *Int. J. Knowledge Management Studies*, Vol. 2, No. 4, pp.460–479.

Biographical notes: Panida Songram is a PhD student in Computer Science at King Mongkut's Institute of Technology Ladkrabang. She is a Lecturer in the Department of Computer Science, Mahasarakham University. Her main research area is data mining.

Veera Boonjing is an Associate Professor in Computer Science, Department of Mathematics and Computer Science at King Mongkut's Institute of Technology Ladkrabang. He received his PhD in Decision Sciences and Engineering Systems from Rensselaer Polytechnic Institute, USA. He teaches Database Systems, Computer Simulation, Programming Fundamentals, Artificial Intelligence, Software Engineering and Mathematics for Computer Science.

His research areas of interest include data mining, database queries, advanced transaction models, multiple pattern matching, case-based reasoning in software engineering and rough set based feature selection.

1 Introduction

Multidimensional sequential pattern mining or MDS pattern mining is an important topic in data mining first introduced by Pinto et al. (2001). The purpose of multidimensional sequential pattern mining or MDS pattern mining is to mine sequential patterns in multidimensional circumstances. MDS patterns can cover more useful information than sequential patterns. For example, a computer device shop may find from its database a sequential pattern $P1$ = (computer, speaker)->printer -> paper holds for 32% of customers. Such a pattern may be popular for the middle-aged customer group but may not be popular for teenagers. Therefore, the pattern $P1$ should be associated with a specific customer group to form an MDS pattern such as (middle, $P1$). This MDS pattern reveals information that the pattern $P1$ appropriated with middle-aged customers.

Many algorithms were proposed to mine MDS patterns. However, all algorithms in MDS pattern mining operate in a bottom-up method, first mine minimal length patterns and then extend one level up in every pass until all maximal length patterns are discovered. Therefore, performance of these algorithms degrades dramatically when mining long sequences or many dimensions or using very low support thresholds. Moreover, a large number of redundant patterns would be generated from those algorithms. These problems are similar to those of itemset pattern mining and sequential pattern mining. To reduce generating the large number of patterns in itemset pattern mining and sequential pattern mining, Galois connection was exploited as a basic concept to find closed patterns. Therefore, the closed itemset pattern mining and the closed sequential pattern mining were proposed to eliminate these problems in itemset pattern mining and sequential pattern mining, respectively.

In this paper, we propose a new method called closed multidimensional sequential pattern mining or CMDS pattern mining. This method is operated based on closed mining. The closed mining would give only patterns which have no a proper superset with the same support. Thus, the set of CMDS patterns is not larger than the set of MDS patterns. Moreover, closed mining assures less information loss because the complete set of MDS patterns can be derived from the complete set of CMDS patterns. All of these are shown in this paper. In addition, an experiment is investigated to determine the effects of mining using two different combinations.

The remainder of this paper is organised as follows. Section 2 mentions related works. In Section 3, we define the basic definitions and properties of CMDS patterns. Sections 4 and 5 introduce CMDS patterns mining method and its correctness, respectively. Experiment results are given in Section 6. Section 7 give conclusion and future work.

2 Related works

In this section, we present three mining topics related to CMDS pattern mining. They are MDS pattern mining, Galois connection, closed itemset pattern mining and closed sequential pattern mining.

2.1 Multidimensional sequential pattern mining

MDS pattern mining is the process of mining one or more dimensions of information in which the order of dimension values is not important, alongside a single dimension of information in which order is important. This topic was introduced by Pinto. PSFP and HYBRID algorithms were proposed to mine MDS patterns (Pinto, 2001). The PSFP mines sequences using PrefixSpan algorithm (Han et al., 2001) followed by mining multidimensional information associated with mined sequences using FP-growth algorithm (Han et al., 2000). The HYBRID mines multidimensional information using BUC algorithm (Beyer and Ramakrishnan, 1999) followed by mining sequences associated with mined multidimensional information using PrefixSpan algorithm. The performance of PSFP is much faster than the HYBRID at low minimum support. When dimensionality is high, the major cost is mining multidimensional information. The PSFP is still faster than the HYBRID because it only mines multidimensional information that occurs with existing sequential patterns. However, the HYBRID is better alternative than the PSFP when data sets are sparse with respect to dimension values present, and dense with respect to the sequential patterns present.

In addition, Pinto proposed two approaches to mine MDS patterns in Pinto et al. (2001);

1 integration of efficient sequential pattern mining and multidimensional analysis methods

2 embedding multidimensional information into sequences and mining the whole set using PrefixSpan algorithm, called Uniseq algorithm.

In the first approach, two algorithms were developed, Dim-Seq and Seq-Dim. Similar to the HYBRID algorithm, the Dim-Seq was developed to mine the patterns from multidimensional information using BUC algorithm first, and then mine sequential patterns from subdatabase using PrefixSpan algorithm. The Seq-Dim takes an opposite way. It is similar to PSFP. The performance study of three algorithms found that Seq-Dim is the fastest algorithm in the most cases; dimensionality is high, number of tuples is high and minimum support is low. In addition, it is the best algorithm when data sets are dense with respect to both dimension value combinations and sequential items. The Uniseq is the most effective when the total number of sequential items plus other dimension values is small. The Dim-Seq outperforms the other methods when datasets are sparse with respect to dimension value combinations, but dense with respect to the sequential patterns present.

2.2 Galois connection

Galois connection is original theoretical framework which formalises the problem of closed pattern mining. The concept of closed patterns is based on the two following functions f and g (Lucchese et al., 2004a,b, 2006).

Let T and I be subset of all the transactions D and subset of all items X appearing in database, respectively

$$f(T) = \{i \in X \mid \forall t \in T, i \in t\} \quad (1)$$

$$g(I) = \{t \in D \mid \forall i \in I, i \in t\} \quad (2)$$

f returns the set of items(itemset) included in all the transactions belonging to T, g returns the set of transactions supporting a given itemset I.

Definition 1: *An itemset I is closed if and only if*

$$c(I) = f(g(I)) = fog(I) = I \quad (3)$$

where the composite function $c = fog$ is called Galois operator or closure operator.

The following properties hold for all I_1, I_2 and T_1, T_2 (Pasquier, et al., 1999, 2000).

$$I_1 \subseteq I_2 \rightarrow g(I_1) \supseteq g(I_2) \quad (4)$$

$$I_1 \subseteq c(I_1) \quad (5)$$

$$c(c(I_1)) = c(I_1) \quad (6)$$

$$I_1 \subseteq I_2 \rightarrow c(I_1) \subseteq c(I_2) \quad (7)$$

$$c'(g(I)) = g(I) \quad (8)$$

$$T_1 \subseteq T_2 \rightarrow f(T_1) \supseteq f(T_2) \quad (9)$$

$$T \subseteq c'(T) \quad (10)$$

$$c'(c'(T_1)) = c'(T_1) \quad (11)$$

$$T_1 \subseteq T_2 \rightarrow c'(T_1) \subseteq c'(T_2) \quad (12)$$

$$c(f(T_1)) = f(T_1) \quad (13)$$

$$T_1 \subseteq g(I_1) \leftrightarrow I_1 \subseteq f(T_1) \quad (14)$$

Example 1: *Consider Table 1 (Lucchese, et al., 2004a,b), the letters A-D represent items in itemset and minimum support is 1. A is not closed because c(A) = f(g(A)) = f({2, 3}) = {A, C, D}. Since c(ACD) = f(g(ACD)) = f({2, 3}) = {A, C, D}, ACD is closed.*

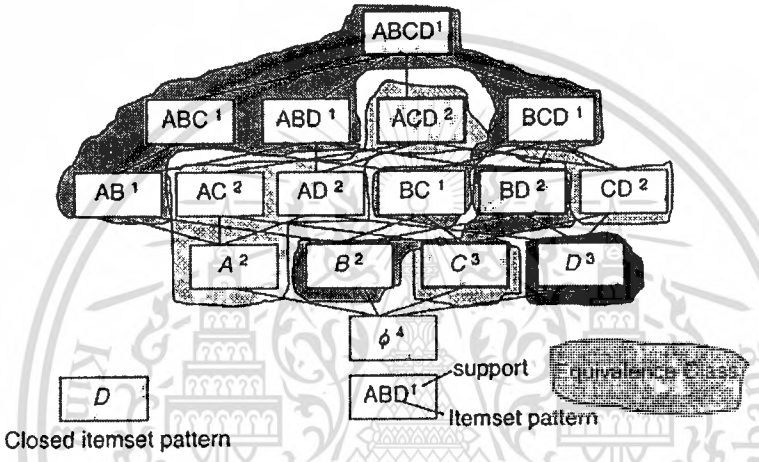
Table 1 Itemset database

TID	*Items*
1	B D
2	A B C D
3	A C D
4	C

From Figure 1 (Lucchese et al., 2004b), the closure operator defines a set of equivalence classes over the lattice of itemset patterns: two itemset patterns belong to the same

equivalence class if and only if they have the same closure. For example, A and ACD are in the same equivalence class, because $c(A) = c(ACD)$. Therefore, the itemset patterns with the same closure are grouped in the same equivalence class. Each equivalence class contains elements sharing the same support, and closed itemset patterns are maximal elements of each equivalence class. Therefore, the number of closed itemset patterns is not more than the number of itemset patterns. In addition, all elements of equivalence class can be derived from the maximal element in the same class.

Figure 1 Lattice of all the itemset patterns



2.3 *closed itemset pattern mining*

Closed itemset pattern mining was proposed to solve the problem of itemset pattern mining. The problem is the generating of a large number of redundant patterns when mining long itemsets or using very low minimum support. It can lead to low performance. Closed itemset pattern mining was introduced to mine only closed itemset patterns, the itemset patterns having no proper superset with the same support. It generates less number of patterns than itemset pattern mining. In spite of fewer patterns generated, closed itemset patterns can derive completely all itemset patterns and their supports. Thus, closed itemset pattern mining can lead not only to a more compact complete result set but also better efficiency. There are many effective algorithms, such as A-CLOSE (Pasquire et al., 1999), CLOSET (Pei et al., 2000), CLOSET + (Wang et al., 2003), TFP (Han et al., 2002), CHARM (Zaki and Hsiao, 2002), GRG (Li et al., 2003) and DCI-CLOSE (Lucchese et al., 2004a) developed for efficient mining of closed itemset patterns. Although these algorithms are dissimilar, the concept of those are based on the same preliminary concepts as follows.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be the complete set of distinct items appearing in a transaction database.

Definition 2: *An itemset* $\delta = (x_1, x_2, \ldots, x_k)$ *is a non-empty subset of I, where k is number of items or length of the itemset, denoted as k-itemset.*

Definition 3: *The number of transactions in database containing itemset δ is called the support of itemset δ, denoted as supp(δ).*

Definition 4: *Itemset $\delta = (x_1, x_2, ..., x_k)$ is a sub-temset of itemset $\lambda = (y_1, y_2, ..., y_l)$ f and only if δ is a subset of λ. λ is a called super-itemset of δ, denoted as $\delta \subset \lambda$. If $\delta = \lambda$, denoted as $\delta \subseteq \lambda$.*

Observation: *If $\delta \subset \lambda$ and* $\text{supp}(\delta) = \text{supp}(\lambda)$, *$\delta$ will be absorbed by λ.*

Definition 5: *The set of itemset patterns FI includes all the itemsets whose support is not less than minimum support, denoted as*

$$FI = \{\delta \mid supp(\delta) \geq min\_support\} \tag{15}$$

Definition 6: *The set of closed itemset patterns CI includes all itemsets in FI which do not have a super-itemset with the same support, denoted as*

$$CI = \{\delta \mid \delta \in FI \wedge (\nexists \lambda \in FI \mid \delta \subset \lambda \wedge supp(\delta) = supp(\lambda))\} \tag{16}$$

From Definitions 5 and 6, we have $CI \subseteq FI$.

Definition 7: *The set of maximal itemset patterns MI includes all itemsets in FI which do not have a super-itemset, denoted as*

$$MI = \{\delta \in FI \mid \nexists \delta' \in FI, \delta \subset \delta'\} \tag{17}$$

Definition 8: *The set of maximal closed itemset patterns MCI includes all itemsets in CI which do not have a super-itemset, denoted as*

$$MCI = \{\lambda \in CI \mid \nexists \lambda' \in CI, \lambda \subset \lambda'\} \tag{18}$$

Example 2: *Consider Table 2 (Li et al., 2003), the letters 1–5 represent items in the itemset and the minimum support is 2. The set of itemset patterns consists of eleven patterns, {(1):3, (2):2, (3):2, (4):3, (5):4, (1, 3):2, (1, 5):2, (2, 4):2, (2, 5):2, (4, 5):3 and (2, 4, 5):2}. Consider pattern (2, 4), it is sub-itemset of pattern (2, 4, 5) and their support are the same. (2, 4) is a redundant pattern because (2, 4) can be derived from (2, 4, 5). Such a redundant pattern would not be mined in closed itemset pattern mining. Therefore, the set of closed itemset patterns contains only six patterns, {(1):3, (5):4, (1, 3):2, (1, 5):2, (4, 5):3 and (2, 4, 5):2}.*

Table 2 Itemset database

Id	*Itemset*		
1.	2	4	5
2.	1	3	–
3.	2	4	5
4.	1	3	5
5.	1	4	5

This example shows that the set of closed itemset patterns is smaller than the set of itemset patterns. In addition, the following properties guarantee that all itemset patterns can be derived from closed itemset patterns, and their supports can be derived from the

support of closed itemset patterns (Lin and Kedem, 1998; Pasquier et al., 1999; Pasquire et al., 1999; Zaki and Hsiao, 2002).

Property 1: All subsets of an itemset pattern are itemset patterns.

Property 2: All supersets of a non itemset pattern are not itemset patterns.

Property 3: All sub-temset of a closed itemset patterns are itemset patterns.

Property 4: The support of an itemset pattern *I* is equal to the support of the smallest closed itemset patterns containing *I*.

Property 5: The set of maximal itemset patterns is identical to the set of maximal closed itemset patterns.

2.4 Closed sequential pattern mining

Sequential pattern mining is similar to itemset pattern mining but it requires an order of itemsets. The problem of generating large redundant patterns occurs with sequential pattern mining. An interesting solution was introduced to eliminate this problem using a similar closed itemset pattern mining concept. It is called closed sequential pattern mining. Many studies have proposed algorithms for mining closed sequential patterns, such as CloSpan (Yan et al., 2003), TSP (Tzvetkov et al., 2003) and BIDE (Wang and Han, 2004). These algorithms are based on the same preliminary concepts as follows.

Definition 9: *A sequence is an ordered list of itemsets. A sequence* α *is denoted by* $\alpha = < a_1 a_2 ... a_n >$, *where* a_i *is an itemset,* $i = 1, ..., n$ *and n is the length of the sequence.*

Definition 10: *A sequence* $\alpha = < a_1 a_2 ... a_n >$ *is a sub-sequence of a sequence* $\beta = < b_1 b_2 ... b_m >$ *if and only if* $\exists i_1, i_2, ..., i_n$, *such that* $1 \le i_1 < i_2 < ... < i_n \le m$ *and* $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, ...,$ *and* $a_n \subseteq b_{i_n}$. β *is called super-sequence of* α, *denoted as* $\alpha \sqsubseteq \beta$. *If* $\alpha \neq \beta$, *denoted as* $\alpha \sqsubset \beta$.

Observation: *If* $\alpha \sqsubset \beta$ *and their supports are the same,* α *will be absorbed by* β.

Definition 11: *The set of sequential patterns FS includes all the sequences whose support is not less than minimum support, denoted as*

$$FS = \{\alpha \mid supp(\alpha) \ge min\_support\} \tag{19}$$

Definition 12: *The set of closed sequential patterns CS includes all sequences in FS which do not have a super-sequence with the same support, denoted as*

$$CS = \{\alpha \mid \alpha \in FS \wedge (\nexists \beta \in FS \mid \alpha \sqsubset \beta \wedge supp(\alpha) = supp(\beta))\} \tag{20}$$

From Definitions 11 and 12, we have $CS \subseteq FS$.

Example 3: *Consider Table 3 (Yan et al., 2003), the letters a, b, d, e and f represent items in sequences and minimum support is 2. The set of sequential patterns consists of 16 patterns, {<a>:3, <b>:2, <d>:2, <e>:3, <f>:2, <ad>:2, <ae>:2, <ab>:2, <fd>:2, <fe>:2, <ea>:3, <eb>:2, <(af)>:2, <(af)d>:2, <(af)e>:2, <eab>:2}. Consider pattern <(af)>, it is sub-sequence of pattern <(af)d> and their support are the same. <(af)> is a redundant pattern because <(af)> can be derived from <(af)d>. Such a redundant*

pattern would not be mined in closed sequential pattern mining. Therefore, the set of closed sequential patterns contains only four patterns, {<(af)d>:2, <(af)e>:2, <ea>:3, <eab>:2}. It is much smaller than set of sequential patterns.

Table 3 Sequence database

Id	*Sequence*
1.	$<(af)\,d\,e\,a>$
2.	$<e\,a\,b>$
3.	$<e\,(abf)\,(bde)>$

3 Basic definitions and properties

3.1 Basic definitions

Given a schema $D = (TID, X_1, X_2,., X_m, S)$ is a multidimensional sequence database; where *TID* is a set of primary keys, $X_1, X_2,., X_m$ is multidimensional information and S is a set of sequences. Let $*$ be any value not belonging to any domain of $X_1, X_2,., X_m$. A multidimensional sequence takes the form (a_1, a_2, a_m, s), where $a_i \in (X_i \cup \{*\})$ for $(1 \le i \le m)$ and s is a sequence. A multidimensional sequence $\alpha = (a_1, a_2, \ldots, a_m, s)$ is said to match a tuple $t = (tid, x_1, x_2, \ldots, x_m, s_t)$ in a multidimensional sequence database if and only if either $a_i = x_i$ or $a_i = *$ for $(1 \le i \le m)$ and the sequence s is a sub-sequence of the sequence s_t.

Definition 13: *The number of transactions in database containing multidimensional sequence α is called the support of multidimensional sequence α, denoted as $supp(\alpha)$.*

Definition 14: *Given a minimum support threshold min_support, the set of complete MDS patterns includes all multidimensional sequences whose support is not less than min_support.*

$$MD = \{\alpha \mid supp(\alpha) \ge min\_support\} \tag{21}$$

Definition 15: *Let $I_m = (a_1, a_2, \ldots, a_m)$ and $I_n = (a_1, a_2, \ldots, a_n)$ be multidimensional information; where m and n is the number of dimensions of the multidimensional information. Let s_l and s_k be sequences; where l and k is number of items in the sequence. A multidimensional sequence $\alpha = (I_m, s_l)$ is a sub-multidimensional sequence of a multidimensional sequence $\beta = (I_n, s_k)$, if one of the following conditions holds.*

$$I_m \subset I_n \text{ and } s_l = s_k \tag{22}$$

$$I_m = I_n \text{ and } s_l \sqsubset s_k \tag{23}$$

$$I_m \subset I_n \text{ and } s_l \sqsubset s_k \tag{24}$$

β is called a super-multidimensional sequence, denoted as $\alpha \subset \beta$.

Observation: *If $supp(\alpha) = supp(\beta)$, then α can be derived from β.*

Example 4: *Consider Table 4 (Pinto, 2001), the letters a–h represent items in the sequences, while the numbers 1–3, 4–6 and 7–9 represent the associated dimension*

values in unordered dimensions D1, D2 and D3, respectively. Suppose the minimum support is 2. Multidimensional sequence (, 6, *, <aa>) is a sub-multidimensional sequence of multidimensional sequence (*, 6, 8, <aa>), because (6)⊂ (6, 8) and <aa>⊏<aba> (*, 6, 8, <aba>) is called a super-multidimensional sequence of (*, 6, *, <aa>). Support of (*, 6, *, <aa>) is equal to support of (*, 6, 8, <aba>). Therefore, (*, 6, *, <aa>) can be derived from (*, 6, 8, <aba>).*

Table 4 Multidimensional sequence database

Id	*D1*	*D2*	*D3*	*Sequence*
1.	1	5	8	$<(b\ d)\ c\ b\ (a\ c)>$
2.	2	6	7	$<(b\ f)\ (c\ e)\ b\ (f\ g)>$
3.	1	6	8	$<(a\ h)\ (b\ f)\ a\ b\ f>$
4.	3	4	9	$<(b\ e)\ (c\ e)\ d>$
5.	2	6	8	$<a\ (b\ d)\ b\ c\ b\ (a\ d\ e)>$

Definition 16: *Given a sequence s, P(s) is the set of all non-empty sub-sequences of s.*

Example 5: *There are three sub-sequences of a sequence <ba>:<b>, <a> and <ba>.*

Definition 17: *Given a multidimensional information I, P(I) is a set of all non-empty sub-multidimensional information of I.*

Example 6: *There are three sub-multidimensional information of multidimensional information (1, 8): (1), (8) and (1, 8).*

Definition 18: *Given a multidimensional sequence $\alpha = (I, s)$, $P(\alpha)$ is the cross product of P(I) and P(s).*

Example 7: *Multidimensional sequence (1, *, 8, <ba>) consists of multidimensional information (1, 8) and sequence <ba>. All sub-multidimensional sequences of (1, *, 8, <ba>) occur by cross product of (1, 8) and <ba>. There are nine sub-multidimensional sequences of (1, *, 8, <ba>):(1, *, *, <a>), (1, *, *, <b>), (1, *, *, <ba>), (*,*, 8, <a>), (*,*, 8, <b>), (*,*, 8, <ba>), (1, *, 8, <a>), (1, *, 8, <b>) and (1, *, 8, <ba>).*

Definition 19: *The set of complete CMDS patterns or CMD is defined as*

$$CMD = \{\alpha \mid \alpha \in MD \wedge (\nexists \beta \in MD \mid \alpha \subset \beta \wedge supp(\alpha) = supp(\beta))\} \quad (25)$$

Example 8: *(*, 6, 8, <ba>) and (*, 6, 8, <aba>) are MDS patterns. (*, 6, 8, <ba>) is a sub-multidimensional sequence of (*, 6, 8, <aba>) with the support value 2. Therefore, (*, 6, 8, <ba>) is not CMDS patterns. The set of CMDS patterns is {(1, *, 8, <ba>):2, (1, *, 8, <bb>):2, (2, 6, *, <bcb>):2, (2, 6, *, <be>):2, (*, 6, *, <(bf)bf>): 2, (*, 6, *, <bb>):3, (*, 6, 8, <aba>):2, (*, 6, 8, <abb>):2, (*,*, 8, <(bd)bc>):2, (*,*, 8, <(bd)ba>):2, (*,*, 8, <cba>):2, (*,*, 8, <bcb>):2, (*,*, 8, <ba>):3, (*,*, 8, <bb>):3}.*

Definition 20: *The set of complete maximal CMDS patterns or MCMD is defined as*

$$MCMD = \{\alpha \mid \alpha \in CMD \wedge (\nexists \beta \in CMD, \alpha \subset \beta)\} \quad (26)$$

Example 9: *From Example 8, a maximal CMDS pattern is a CMDS pattern having no a proper super-multidimensional sequence. Thus, the set of complete maximal CMDS patterns is {(1, *, 8, <ba>), (1, *, 8, <bb>), (2, 6, *, <bcb>), (2, 6, *, <be>), (*, 6, *, <(bf)bf>), (*, 6, 8, <aba>), (*, 6, 8, <abb>), (*,*, 8, <(bd)bc>), (*,*, 8, <(bd)ba>), (*,*, 8, <cba>), (*,*, 8, <bcb>)}.*

Definition 21: *The set of complete maximal MDS patterns or MMD is defined as*

$$MMD = \{\alpha \mid \alpha \in MD \wedge (\nexists \beta \in MD, \alpha \subset \beta)\}. \quad (27)$$

Example 10: *The set of complete maximal MDS patterns consists of the patterns in MD which has no proper super-multidimensional sequences. Therefore, it is {(1, *, 8, <ba>), (1, *, 8, <bb>), (2, 6, *, <bcb>), (2, 6, *, <be>), (*, 6, *, <(bf)bf>), (*, 6, 8, <aba>), (*, 6, 8, <abb>), (*,*, 8, <(bd)bc>), (*,*, 8, <(bd)ba>), (*,*, 8, <cba>), (*,*, 8, <bcb>)}.*

3.2 Properties

The following CMDS properties guarantee that,

1 all CMDS patterns are discovered

2 all MDS patterns and their supports can be derived from CMDS patterns.

Property 1: All sub-multidimensional sequences of an MDS pattern are MDS patterns.

Proof: Suppose $M \in MD$ and $M' \subset M$

$M' \subset M \rightarrow supp(M') \geq supp(M)$

But $M \in MD$: $supp(M) \geq min\_support$

So, $supp(M') \geq min\_support$.

Therefore, $M' \in MD$. □

Example 11: *All sub-multidimensional sequences of a MDS pattern (<(bd)>, *, *, 8):2 consist of (<b>, *, *, 8):3, (<d>, *, *, 8):2 and (<(bd)>, *, *, 8):2. Their supports are not lower than the minimum support, so they are MDS patterns.*

Property 2: All super-multidimensional sequences of a non-MDS pattern are not MDS patterns.

Proof: Suppose $M \notin MD$ and $M \subset M'$

$M \subset M' \rightarrow supp(M') \leq supp(M)$

But $M \notin MD$: $supp(M) < min\_support$

So, $supp(M') < min\_support$.

Therefore, $M' \notin MD$. □

Example 12: *Multidimensional sequence (2, 6, *, <(bd)bcb(ade)>):1 is not an MDS pattern. All super-multidimensional sequences of this pattern consist of (2, 6, 8,*

*<(bd)bcb(ade)>):1, (2, 6, *, <a(bd)bcb(ade)>):1 and (2, 8, 6, <a(bd)bcb(ade)>):1. They are not MDS patterns because their supports are lower than minimum support.*

Property 3: All sub-multidimensional sequences of a CMDS pattern are MDS patterns.

Proof: Suppose $M \in CMD$ and $M' \subset M$

$M \in CMD$: $M \in MD$

So $M \in MD$ and $M' \subset M$

Therefore, $M' \in MD$. □

Example 13: *Multidimensional sequence (1, *, 8, <ba>):2 is a CMDS pattern. It consists of eight sub-multidimensional sequences, (1, *, *, <a>):2, (1, *, *, <b>):2, (1, *, *, <ba>):2, (*,*, 8, <a>):3, (*,*, 8, <b>):3, (*,*, 8, <ba>):3, (1, *, 8, <a>):2, (1, *, 8, <b>):2 and (1, *, 8, <ba>):2, whose supports are not lower than minimum support. Therefore, all sub-multidimensional sequences of (1, *, 8, <ba>) are MDS patterns.*

Property 4: The set of maximal MDS patterns *MMD* is identical to the set of maximal CMDS patterns *MCMD*.

Proof: To show $MMD = MCMD$, we have to show that $MMD \subseteq MCMD$ and $MCMD \subseteq MMD$.

Case 1: $MMD \subseteq MCMD$.

Let $M \in MMD$. We have to show that $M \in MCMD$; that is, show that $M \in CMD$ and there is no proper super-multidimensional sequence of M in MD with the same support.

Part 1: We show that $M \in CMD$.

Since $M \in MMD$, so $M \in MD$. And we know that there is no proper super-multidimensional sequence of M in MD according to definition 21. It shows there is no proper super-multidimensional sequence of M in MD with the same support. Thus, we can conclude that $M \in CMD$ according to Definition 19.

Part 2: We show that there is no proper super-multidimensional sequence of M in CMD.

Because $M \in MMD$, there is no proper super-multidimensional sequence of M in MD. Therefore, there is no proper super-multidimensional sequence of M in CMD.

From part 1 and 2, we can conclude that $M \in MCMD$ according to Definition 20.

Case 2: $MCMD \subseteq MMD$

Let $M \in MCMD$. We have to show that $M \in MMD$; that is, show that $M \in MD$ and there is no proper super-multidimensional sequence of M in MD.

Part 1: We show that $M \in MD$.

Because $MCMD \subseteq CMD$ and $CMD \subseteq MD$, so $M \in MD$

Part 2: We show that there is no proper super-multidimensional sequence of M in MD

Since $M \in MCMD$, there is no proper super-multidimensional sequence of M in CMD. Assume there is proper super-multidimensional sequence of M in MD, call it M'. M' has a super-sequence in CMD, call it M''. So, we know M'' is a super-sequence of M', and M' is a super-sequence of M. Therefore, M'' is a proper super-sequence of M in CMD. This contradicts $M \in MCMD$.

From parts 1 and 2, we can conclude that $M \in MMD$.

From cases 1 and 2, we can conclude that $MMD = MCMD$. □

Example 14: *From Examples 9 and 10, we found that the set of maximal MDS patterns and the set of maximal CMDS patterns are the same.*

Property 5: The support of an MDS pattern α is equal to the support of the smallest CMDS pattern containing α.

Proof: We have to show that $supp(\alpha) = supp(c(\alpha))$.

Let $supp(\alpha) = |g(\alpha)|.c(\alpha)$ be closure of α. It means that α can be derived from $c(\alpha)$. Since $c(\alpha)$ is closed and $c'(g(\alpha)) = g(\alpha)$ according to Property 5 of Galois connection. We have $supp(c(\alpha)) = |g(c(\alpha))| = |c'(g(\alpha))| = |g(\alpha)| = supp(\alpha)$. Therefore, $supp(\alpha) = supp(c(\alpha))$. □

Example 15: *The support of an MDS pattern (*, 6, *, <ba>) is equal to the support of a CMDS pattern (*, 6, 8, <aba>), which is the smallest CMDS pattern containing (*, 6, *, <ba>).*

4 Closed multidimensional sequential pattern mining method

The method of CMDS pattern mining consists of two major steps:

1 combine closed itemset pattern mining with closed sequential pattern mining

2 eliminate redundant patterns from the result of the first step.

4.1 Combination of closed itemset pattern mining with closed sequential pattern mining

Let $CI = \{x_1, x_2, \ldots, x_h\}$ be a set of closed itemset patterns, $CS = \{y_1, y_2, \ldots, y_k\}$ be a set of closed sequential patterns.

We integrate closed sequential pattern mining and closed itemset pattern mining using the following two approaches.

In the first approach, *CI* is firstly mined from multidimensional information using closed itemset pattern mining. Then *CCMD* (the set of candidate CMDS patterns) for each closed itemset pattern in *CI* is obtained, as shown below, by mining *CS* from sequences using closed sequential pattern mining. The union of all *CCMDs* is called $CCMD_{CI}$.

$$CCMD_1 = x_1 \times CS_1$$

$$CCMD_2 = x_2 \times CS_2$$

...

...

$$CCMD_h = x_h \times CS_h$$

In the second approach, *CS* is firstly mined from sequences using closed sequential pattern mining. Then *CCMD* for each closed sequential pattern in *CS* is obtained, as

shown below, by mining *CI* from multidimensional information using closed itemset pattern mining. The union of all *CCMDs* is called $CCMD_{CS}$.

$$CCMD_1 = y_1 \times CI_1$$

$$CCMD_2 = y_2 \times CI_2$$

...

...

$$CCMD_k = y_k \times CI_k$$

Note that both $CCMD_{CS}$ and $CCMD_{CI}$ are not necessarily sets of CMDS patterns as shown in the following example.

Example 16: *The mining followed by the second approach gives the patterns, {(2, 6, *, <bcb>):2, (2, 6, *, <be>):2, (2, 6, *, <cb>):2, (1, *, 8, <ba>):2, (1, *, 8, <bb>):2, (*,*, 8, <ba>):3, (*,*, 8, <bb>):3, (*,*, 8, <aba>):2, (*,*, 8, <abb>):2, (*,*, 8, <cba>):2, (*,*, 8, <(bd)ba>):2, (*,*, 8, <(bd)bc>):2, (*,*, 8, <bcb>):2, (*, 6, *, <bb>):3, (*, 6, *, <be>):2, (*, 6, *, <aba>):2, (*, 6, *, <abb>):2, (*, 6, *, <bcb>):2, (*, 6, *, <(bf)bf>):2, (*, 6, 8, <aba>):2, (*, 6, 8, <abb>):2}. From these patterns, pattern (2, 6, *, <be>) is a super-multidimensional sequence of (*, 6, *, <be>), and its support is equal to the support of (*, 6, *, <be>). (*, 6, *, <be>) is absorbed by (2, 6, *, <be>). Thus, (*, 6, *, <be>) is a redundant pattern. We need to eliminate such a pattern.*

4.2 Elimination of redundancy

Given $c, c' \in CCMD$, if $c' \subset c$ and $supp(c') = supp(c)$, then a candidate CMDS pattern c' is redundant and can be eliminated. A set of CMDS patterns containing no redundant element will be denoted as *CMD*.

Example 17: *We know that the supports of both (*, *, 8, <bc>) and (*,*, 8, <(bd)bc>) are 2 and (*, *, 8, <bc>)⊂ (*,*, 8, <(bd)bc>). Then, the pattern (*,*, 8, <bc>) is redundant and can be eliminated.*

5 Correctness

The following theorems show correctness of the proposed method of CMDS pattern mining.

Theorem 1: *$CMD \subseteq MD$, where CMD and MD are sets of CMDS patterns and MDS patterns, respectively.*

Proof: Assume $(y, x) \in CCMD$, where $y \in CS$ and $x \in CI$. We know that $CI \subseteq FI$, where *CI* is a set of closed itemset patterns, *FI* is a set of itemset patterns; and $CS \subseteq FS$, where *CS* is a set of closed sequential patterns and *FS* is a set of sequential patterns. Therefore, $y \in FS$ and $x \in FI$. This implies that $(y, x) \in MD$. Therefore, $CCMD \subseteq MD$. From the proposed method, we know that $CMD \subseteq CCMD$. Therefore, $CMD \subseteq MD$. □

Theorem 1 implies that the number of CMDS patterns mined from a database is not larger than the number of MDS patterns mined from the same database. This is denoted by $|CMD| \leq |MD|$.

Theorem 2: *The set of MDS patterns can be generated from the set of CMDS patterns.*

Proof: Based on Property 4, all MDS patterns can be derived from the maximal CMDS patterns. Based on Property 5, the support of each MDS pattern can be derived from the support of CMDS pattern. We can conclude that the set of MDS patterns can be generated from the set of CMDS patterns.

Theorem 3: $(s \sqsubset s') \wedge (supp(s) = supp(s')) \rightarrow I_s = I_{s'}$, *where* $I_{s'}$ *and* I_s *are sets of transaction ids of multidimensional information associated with a sequence s′ and s, respectively.*

Proof: Assume ($s \sqsubset s'$), $supp(s) = supp(s')$ and $I_s \neq I_{s'}$, so there are two cases to be considered:

Case 1: $|I_s| \neq |I_{s'}|$. We have $supp(s) \neq supp(s')$. This contradicts our assumption.

Case 2: $|I_s| = |I_{s'}|$. We have $s \not\sqsubset s'$. This contradicts our assumption.

From Cases 1 and 2, we can conclude that $(s \sqsubset s') \wedge (supp(s) = supp(s')) \rightarrow I_s = I_{s'}$ □

Theorem 3 implies that multidimensional information associated with sub-sequences of s' is the same as multidimensional information associated with s' if the support of sub-sequences of s' is equal to the support of s'. Therefore, closed itemset pattern mining on multidimensional information mines only multidimensional information associated with mined closed sequential patterns.

Example 18: *Sequence <cb> is a sub-sequence of <bcb> with the same support as 3. The multidimensional information associated with a sequence <bcb> is found in transaction id 1, 2 and 5 as seen in table 4. Moreover, the multidimensional information associated with sequence <cb> is also found in transaction id 1, 2 and 5. Therefore, the multidimensional information associated with sequence <bcb> and the multidimensional information associated with sequence <cb> are the same as shown in Table 5.*

Table 5 Multidimensional information associated with sequence <*bcb*>

Id	*D1*	*D2*	*D3*
1.	1	5	8
2.	2	6	7
5.	2	6	8

Theorem 4: $(I \subset I') \wedge (supp(I) = supp(I')) \rightarrow S_I = S_{I'}$, *where* $S_{I'}$ *and* S_I *are sets of transaction ids of sequences associated with multidimensional information I′ and I, respectively.*

Proof: Assume ($I \subset I'$), $supp(I) = supp(I')$, and $S_I \neq S_{I'}$ so there are two cases to be considered:

Case 1: $|S_I| \neq |S_{I'}|$. We have $supp(I) \neq supp(I')$. This contradicts our assumption.

Case 2: $|S_I| = |S_{I'}|$. We have $I \subset I'$. This contradicts our assumption.

From Cases 1 and 2, we can conclude that $(I \subset I') \wedge (supp(I) = supp(I')) \rightarrow S_I = S_{I'}$ □

Theorem 4 implies that sequences associated with sub-multidimensional information of I' is the same as sequences associated with I' if the support of sub-multidimensional information of I' is equal to the support of I'. Therefore, closed sequential pattern mining on sequences mines only sequences associated with mined closed itemset patterns.

Example 19: *From Table 4, multidimensional information (2) is a sub-multidimensional information of (2, 6) and the support of multidimensional information (2) is equal to the support of multidimensional information (2, 6) as 2. The sequences associated with multidimensional information (2, 6) are found in transaction id 2 and 5 as given in Table 6. Furthermore, the sequences associated with multidimensional information (2) are also found in transaction id 2 and 5. Thus, the sequences associated with multidimensional information (2) and (2, 6) are the same.*

Table 6 Sequences associated with multidimensional information (2, 6)

Id	*Sequence*
2.	$<(b\ f)(c\ e)\ b\ (f\ g)>$
5.	$<a(b\ d)\ b\ c\ b\ (a\ d\ e)>$

In summary, Theorems 1 and 2 guarantee that the number of CMDS patterns is not larger than the number of MDS patterns and CMDS patterns cover all MDS patterns, respectively. Theorem 3 guarantees that mining using closed itemset pattern mining on multidimensional information mines only multidimensional information associated with mined closed sequential patterns. Theorem 4 guarantees that mining using closed sequential pattern mining on sequences mines only sequences associated with mined closed itemset patterns.

6 Experiment results

In this section, we investigate experiments in order to compare the number of candidate patterns. Two algorithms were developed for this purpose, CIS and CSI. Both algorithms apply CLOSET algorithm and Clospan algorithm as a closed itemset pattern mining algorithm, and a closed sequential pattern mining algorithm, respectively. Since candidate patterns are only considered in these experiments and all closed itemset and closed sequential algorithms with the same data set surely give the same results, we can use whatever closed itemset mining algorithm and closed sequential pattern mining algorithm in our algorithms.

Algorithm 1: (CIS- mining closed itemset patterns before closed sequential patterns)
Input: Multidimensional sequence database *SDB* and minimum support threshold *min_support*.
Output: The set of candidate CMDS patterns
Method: First mine itemset patterns in *SDB* using CLOSET algorithm. Sequences corresponding to each closed itemset pattern obtained are mined by using CloSpan algorithm.

Algorithm 2 (CSI – mining closed sequential patterns before closed itemset patterns)
Input and output: same as algorithm1.

Method: First mine closed sequential patterns in *SDB* using CloSpan algorithm. Multidimensional information corresponding to each closed sequential pattern obtained is mined by using CLOSET algorithm.

Both algorithms, CIS and CSI, were tested with synthetic data sets, C20N10T2.5S8I1.25 and C10N10T2.5S8I1.25. Synthetic data sets consist of sequences and multidimensional information. Sequences are generated using the IBM generator. In these sequences, the number of itemsets is set to 10, 000 while the number of sequences is set to 10, 000 for C10N10T2.5S8I1.25 and 20, 000 for C20N10T2.5S8I1.25. The average number of items within each element is 2.5. The average number of elements in one sequence is 8. Dimensional information is generated by a program which randomises values in each dimension. The value of each dimension depended on the number of dimensions, for example, 2_2 represents 2 dimensions with 2 values each (1–2).

All experiments were performed on PCs with Intel Pentium III processor 1.00 GHz, 256 MB SD-RAM and 20 GB hard-disk. Algorithms were coded in Visual Studio C + + Version 6.0 and run on Window XP Professional Version 2003.

Figures 2 and 3 plot the number of candidate CMDS patterns of the algorithms over the number of dimensions. From Figures 2 and 3, CIS gives fewer candidate CMDS patterns than CSI when the number of dimensions is not more than three. But when the dimensionality is high, CSI gives fewer candidate CMDS patterns than CIS.

Figure 2 Number of candidate CMDS patterns over dimensionality of data set C10N10T2.5S8I1.25 (see online version for colours)

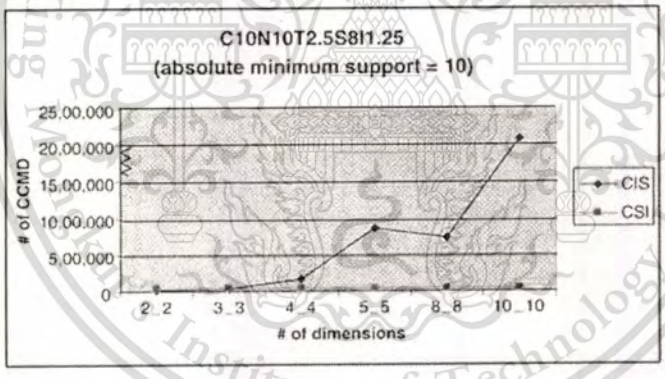
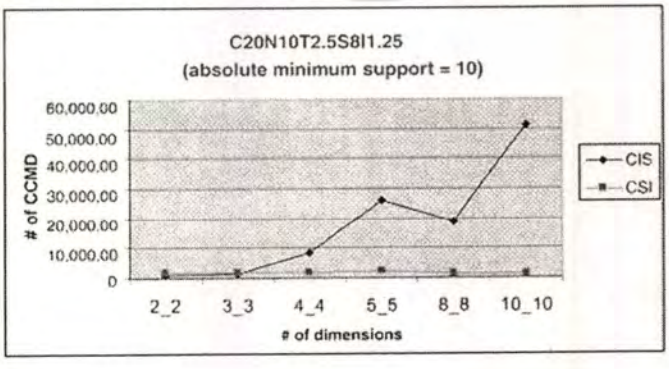


Figure 3 Number of candidate CMDS patterns over dimensionality of data set C20N10T2.5S8I1.25 (see online version for colours)



Figures 4–7 show the numbers of candidate CMDS patterns of both algorithms over minimum support. At minimum support 5, 10 and 20; CIS gives fewer patterns than CSI when the dimension is set to 2 or 3. But, CSI gives fewer patterns than CIS when the dimension is set to 4 or 10.

Figure 4 Number of candidate CMDS patterns over minimum support of data set C10N10T2.5S8I1.25 with 2 dimensions (see online version for colours)

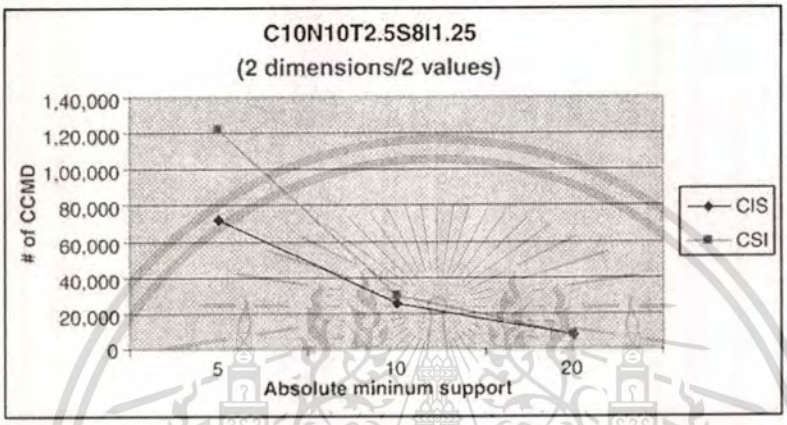
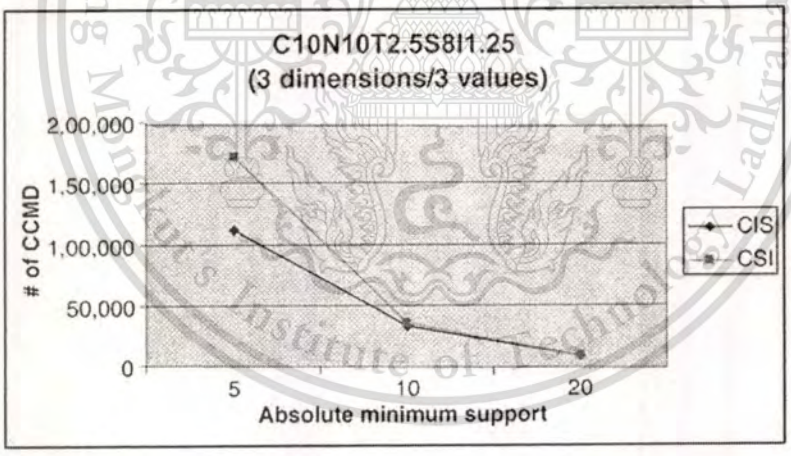


Figure 5 Number of candidate CMDS patterns over minimum support of data set C10N10T2.5S8I1.25 with 3 dimensions (see online version for colours)



In summary, CIS gives fewer candidate CMDS patterns than CSI when dimensionality is low because it only mines closed sequential patterns that occur with few closed itemset patterns. When dimensionality is high, CIS firstly generates many closed itemset patterns, and then generates a lot of closed sequential patterns of each closed itemset pattern. Therefore, a large number of candidate CMDS patterns are generated by CIS. Meanwhile, CSI only generates closed itemset patterns that occur with few closed sequential patterns. Thus, CSI gives fewer candidate CMDS patterns than CIS.

Figure 6 Number of candidate CMDS patterns over minimum support of data set C10N10T2.5S8I1.25 with 4 dimensions (see online version for colours)

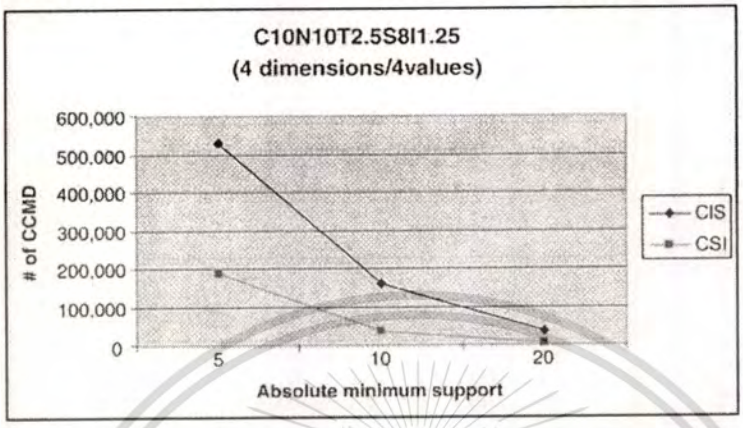
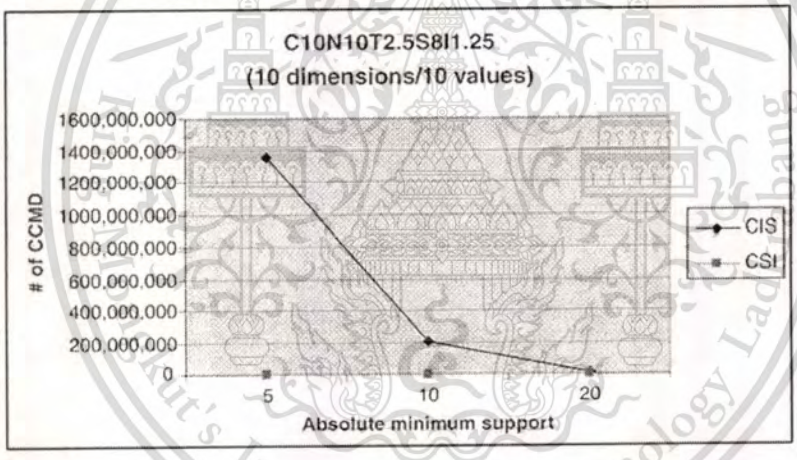


Figure 7 Number of candidate CMDS patterns over minimum support of data set C10N10T2.5S8I1.25 with 10 dimensions (see online version for colours)



7 Conclusion and future work

In this study, we propose a new method to mine CMDS patterns based on closed itemset pattern mining and closed sequential pattern mining. The new method consists of two major steps;

1 combination of closed itemset pattern mining with closed sequential pattern mining

2 elimination of redundant patterns.

The second step is based on the result that patterns obtained from the first step are not necessarily CMDS patterns. Based on this method, we show that

1 the number of CMDS patterns is not larger than the number of MDS patterns

2 the set of CMDS patterns covers the set of MDS patterns

3 mining using closed itemset pattern mining on multidimensional information mines only multidimensional information associated with mined closed sequential patterns, and mining using closed sequential pattern mining on sequences mines only sequences associated with mined closed itemset patterns.

We also investigated experiments to determine factors affecting the number of candidate patterns. Generated data sets were tested with two algorithms which were based on two combinations:

1 mining of closed itemset patterns followed by closed sequential patterns

2 mining of closed sequential patterns followed by closed itemset patterns.

CLOSET and CloSpan algorithms were exploited in our implementation as closed itemset pattern mining and closed sequential pattern mining algorithms, respectively. The results show that the first approach is suitable in the first step of CMDS pattern mining on the data sets with few dimensions. For the data sets with many dimensions, the second approach is the better alternative. The experiment results imply that any efficient method for mining CMDS patterns must consist of two separated submethods handling two different classes of data sets based on their dimensions. Therefore, our future work is development of such efficient methods including an extreme method generating non-redundant CMDS patterns.

References

Afshar, R. (2000) 'Mining frequent max and closed sequential patterns', MSc Thesis, University of Alberta, Alberta.

Beyer, K. and Ramakrishnan, R. (1999) 'Bottom-up computation of sparse and iceberg cubes', *ACM-SIGMOD International Conference Management of Data (SIGMOD'99)*, Philadelphia, Pennsylvnia, USA, pp.359–370.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. and Hsu, M-C. (2001) 'PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth', *International Conference on Data Engineering (ICDE'01)*, Heidelberg, Germany, pp.215–224.

Han, J., Pei, J. and Yin, Y. (2000) 'Mining frequent patterns without candidate generation', *ACM-SIGMOD International Conference Management of Data (SIGMOD'00)*, Dallas, TX, pp.1–12.

Han, J., Wang, J., Lu, Y. and Tzvetkov, P. (2002) 'Mining top-k frequent closed patterns without minimum support', *International Conference on Data Mining (ICDM'02)*, Maebahi, Japan, pp.211–218.

Li, L., Zhai, D. and Jin, F. (2003) 'GRG: an efficient method for association rules mining on frequent closed itemsets', *IEEE International Symposium on Intelligent Control (ISIC'03)*, Houston, Texas, pp.854–859.

Lin, D. and Kedem, Z. (1998) 'Pincer-Search: a new algorithm for discovering the maximum frequent set', *The 6th International Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, pp.105–119.

Lucchese, C., Orlando, S. and Perego, R. (2004a) 'DCI-CLOSED: a fast and memory efficient algorithm to mine frequent closed itemsets', *IEEE ICDM workshop on Frequent Itemset Mininig Implemention (FIMI'04)*, Brighton, UK, Vol. 126.

Lucchese, C., Orlando, S. and Perego, R. (2004b) 'Mining frequent closed itemsets without duplicates generation', *Technical Report*.

Lucchese, C., Orlando, S. and Perego, R. (2006) 'Fast and memory efficient mining of frequent closed itemsets', *The Journal of IEEE transaction on Knowledge and Data Engineering*, Vol. 18, No. 1, pp.21–36.

Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L. (1999) 'Efficient mining of association rules using closed itemset lattices', *The Journal of Information Systems*, Vol. 24, No.1, pp.25–46.

Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L. (2000) 'Mining based for associate rules using dalois closed sets', *In Research Report*, Computer Science Department, Nice Sophia Antipolis University.

Pasquire, N., Bastide, Y., Taovil, R. and Lakhal, L. (1999) 'Discovering frequent closed itemsets for association rules', *The 7th International Conference on Database Theory (ICDT'99)*, Jerusalem, Israel, pp.398–416.

Pei, J., Han, J. and Mao, R. (2000) 'CLOSET: an efficient algorithm for mining frequent closed itemsets', *ACM-SIGMOD International Workshop on Data Mining and Knowledge Discovery (DMKD'00)*, pp.21–30.

Pinto, H. (2001) 'Multi-dimensional sequential pattern mining', MSc Thesis, Simon Fraser University, Canada.

Pinto, H., Han, J., Pei, J., Wang, K., Chen, Q. and Dayal, U. (2001) 'Multi-dimensional sequential pattern mining', *2001 ACM CIKM International Conference on Information and Knowledge Management(ACM'01)*, Atlanto, Georgia, USA, pp.81–88.

Tzvetkov, P., Yan, X. and Han, J. (2003) 'TSP: mining top-k closed sequential patterns', *The 3rd IEEE International Conference on Data Mining (ICDM'03)*, Melbourne, Florida, USA, pp.347–354.

Wang, J. and Han, J. (2004) 'BIDE: efficient mining of frequent closed sequences', *The 20th International Conference on Data Engineering (ICDE'04)*, Boston, MA, USA, pp.79–90.

Wang, J., Han, J. and Pei, J. (2003) 'CLOSET + : searching for the best strategies for mining frequent closed itemsets', *The 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, Washington, DC, pp.236–245.

Yan, X., Han, J. and Afshar, R. (2003) 'CloSpan: mining closed sequential patterns in large database', *SIAM International Conference Data Mining (SDM'03)*, San Francisco, CA, pp.166–177.

Zaki, M.J. and Hsiao, C.J. (2002) 'CHARM: an efficient algorithm for closed itemsets mining', *SAIM International Conference on Data Mining (SDM'02)*, Arlington, VA, pp.457–473.

Efficient Algorithms for Mining Closed Multidimensional Sequential Patterns

Veera Boonjing[1] Panida Songram[1,2]
[1]*Software Systems Engineering Laboratory,*
Department of Mathematics and Computer Science, Faculty of Science,
King Mongkut's Institute of Technology Ladkrabang,
Bangkok, Thailand, kbveera@kmitl.ac.th
[2]*Department of Computer Science, Faculty of Informatics,*
Mahasarakham University, Mahasrakham, Thailand, panida.s@msu.ac.th

Abstract

A combination of closed sequential pattern mining and closed itemset pattern mining was proposed to mine closed multidimensional sequential patterns. There are two ways for this combination; (1) mining closed itemset patterns from multidimensional information followed by mining closed sequential patterns from sequences associated with closed itemset patterns, and (2) mining closed sequential patterns from sequences followed by mining closed itemset patterns from multidimensional information associated with closed sequential patterns. In the first way the major cost is mining all sequences associated with closed itemset patterns. A similar problem occurs with the second way, the major cost is mining all multidimensional information associated with closed sequential patterns. Therefore, this paper proposes two new combinations that don't need to mine all sequences in the first combination, and all multidimensional information in the second combination. Both combinations can be effected by exploiting two concepts. In the first combination, any closed sequential patterns associated with a closed itemset patten l can be found from a set of closed sequential patterns associated with a closed itemset pattern l' if $l \supset l'$. In the second combination, any closed itemset patterns associated with a closed sequential pattern s can be found from a set of closed sequential patterns associated with a closed itemset pattern s' if $s \supset s'$.

1. Introduction

Multidimensional sequential pattern mining is an important topic in data mining. Many researchers have studied this topic but different propositions. Yu et at. [1] proposed to mine sequential patterns from multidimensional sequence data containing several dimensions. AprioriMD and PrefixMDSpan were proposed in this work by applying the Apriori and PrefixSpan algorithm, respectively. Plantevit et al. [4] consider patterns containing several dimensions combined over time. M2SP algorithm was proposed to mine the patterns. However, the number of the patterns found from a database can be huge. To reduce the amount of discovered patterns, MD-Closed algorithm [5] was proposed by applying closed mining concept, only patterns having no a proper superset with the same support are mined. Pinto et al. proposed to mine multidimensional information alongside sequence. The PSFP and HYBRID algorithms were proposed to mine the patterns in [2]. PSFP mines sequences using the PrefixSpan algorithm followed by mining multidimensional information associated with mined sequences using the FP-growth algorithm. HYBRID mines multidimensional information using the BUC algorithm followed by mining sequences associated with mined multidimensional information using the PrefixSpan algorithm. In addition, Dim-Seq, Seq-Dim and Uniseq were proposed in [3]. Dim-Seq and Seq-Dim are similar to HYBRID and PSFP, respectively. Uniseq embeds multidimensional information into sequences and mining the whole set using the PrefixSpan algorithm. Since PSFP, HYBRID, Dim-Seq, Seq-Dim may generate the large number of redundant patterns, closed multidimensional sequential pattern mining [6] was proposed to solve this problem. This work combines closed itemset pattern mining and closed sequential pattern mining.

The combination has to mine two parts, sequences and multidimensional information. If sequences are mined first, the main cost is mining all multidimensional information associated with mined

Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)
0-7695-2874-0/07 $25.00 © 2007 IEEE

sequences. On the other hand, the main cost is mining all sequences associated with mined multidimensional information if multidimensional information is mined first. This paper proposes novel efficient algorithms to reduce these costs.

The remainder of the paper is organized as follows. In Section 2, we define the basic definitions of closed multidimensional sequential patterns. New algorithms for mining closed multidimensional sequential patterns as well as their correctness are given in Section 3. Section 4 gives conclusion.

2. Basic definitions

A schema $D = (T, X_1, X_2, ..., X_m, S)$ is a multidimensional sequence database; where T is a primary key and $X_1, X_2, ..., X_m$ is multidimensional information and S is sequences. Let * be any value not belonging to any domain of $X_1, X_2, ..., X_m$. A multidimensional sequence takes the form $(a_1, a_2, ..., a_m, s_k)$, where $a_i \in (X_i \cup \{*\})$ for $(1 \le i \le m)$ and s_k is a sequence. A multidimensional sequence $\alpha = (a_1, a_2, ..., a_m, s_k)$ is said to match with a tuple $t = (tid, x_1, x_2, ..., x_m, s_t)$ in a multidimensional sequence database if and only if either $a_i = x_i$ or $a_i = *$ for $(1 \le i \le m)$ and the sequence s_k is a sub-sequence of the sequence s_t, denoted as $s_k \sqsubseteq s_t$. (if $s_k \ne s_t$, denoted as $s_k \sqsubset s_t$). The number of tuples in the database matching a multidimensional sequence α is called the support of α, denoted as $supp(\alpha)$. Given a minimum support threshold *min_support*, the multidimensional sequence α is called a multidimensional sequential pattern if and only if $supp(\alpha) \ge min\_support$. A multidimensional sequence $\beta = (b_1, b_2, ..., b_n, s_l)$ is a sub-multidimensional sequence of the multidimensional sequence α if one of the following conditions holds; (1) $(b_1, b_2, ..., b_n) \subset (a_1, a_2, ..., a_m)$ and $s_l = s_k$, (2) $(b_1, b_2, ..., b_n) = (a_1, a_2, ..., a_m)$ and $s_l \sqsubset s_k$, and (3) $(b_1, b_2, ..., b_n) \subset (a_1, a_2, ..., a_m)$ and $s_l \sqsubset s_k$. The multidimensional sequence α is called a closed multidimensional sequential pattern if and only if $supp(\alpha) \ge min\_support$ and no super-multidimensional sequence with the same support.

Let $T = \{t_1, t_2, ..., t_n\}$ be the set of all transaction ids in a database, $X \subset Y$, where X and Y are multidimensional information or a sequence and $P_x = \{p_1, p_2, ..., p_l\}$ be a set of closed patterns associated with X.

Definition 1. The set of all transactions supporting a pattern X is defined as $t(X) = \{t \in T \mid \forall x \in X, x \in t\}$.

Observation: $|t(X)|$ is the support of X

Example 1. Consider table 1, the letters a – h represent items in the sequences, while the numbers 1 – 3, 4 – 6 and 7 – 9 represent the associated dimension values in unordered dimensions D1, D2 and D3, respectively. Multidimensional information (6, 8) is contained in transaction ids 3 and 5. So the set of all transactions supporting (6, 8) is {3, 5}.

Table 1. Multidimensional sequence database

Id	D1	D2	D3	Sequence
1.	1	5	8	<(bd)cb(ac)>
2.	2	6	7	<(bf)(ce)b(fg)>
3.	1	6	8	<(ah)(bf)abf>
4.	3	4	9	<(be)(ce)d>
5.	2	6	8	<a(bd)bcb(ade)>

Definition 2. The set of different transaction ids between pattern X and pattern Y is defined as $t_{X-Y} = t(X) - t(Y)$.

Example 2. Multidimensional information (6, 8) is super-multidimensional information of multidimensional information (6). The set of transaction ids containing (6, 8) is t(6, 8) = {3, 5} and the set of transaction ids containing (6) is t(6) = {2, 3, 5}. Therefore, the set of different transaction ids between (6) and (6, 8) is t(6)-t(6, 8) = {2, 3, 5}-{3, 5} = {2}.

Definition 3. The support of patterns associated with Y is denoted as $supp(p_y) = supp(p_x) - supp(p_{x-y})$, where $p_x = p_{x-y}$, p_{x-y} is any patterns in P_x occurring in different transaction t_{x-y}.

Example 3. The support of a sequential pattern <bb> associated with <6> is 3. The support of a sequential pattern <bb> associated with <8,6> is supp($<bb>_{(6)}$)-supp($<bb>_{(6)-(8,6)}$) = 3-1 =2.

Definition 4. The set of patterns associated with Y is denoted as follows.

$$P_y = \{p_x \in P_x \mid supp(p_x) - supp(p_{x-y}) \ge min\_support \wedge (\nexists p' \in P_x \mid p_x \subset p' \wedge supp(p_x) = supp(p'))\}$$

Example 4. A closed sequential pattern <b> is contained in transaction ids {1, 2, 3, 4, 5}. (6):3, (8):3, (1, 8):2, (2, 6):2, (6, 8):2 are closed itemset patterns associated with <b>. Closed sequential pattern <bc> is contained in transaction ids {1, 2, 4, 5}. <b> is a sub-sequence of <bc>. The different transaction id between <b> and <bc> is 3. There are four patterns of $P_{<b>}$ occurring in transaction id 3, (6):1, (8):1, (1, 8):1, (6, 8):1. Their supports are operated with support of closed itemset associated with <b>, supp(6) = 3-1 = 2, supp(8)=3-1=2, supp(1, 8)=2-1=1, supp(2, 6)=2-0=2, supp(6, 8)=2-1=1. (1, 8) and (6, 8) are not closed itemset patterns associated with <bc> because their supports are less than the minimum support threshold.

Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)
0-7695-2874-0/07 $25.00 © 2007 IEEE

(6) is not a closed itemset pattern associated with <bc> because (6) is a subset of <2,6> with the same support. So the closed itemset patterns associated with <bc> are (8):2 and (2, 6):2.

3. Algorithms

3.1. Existing algorithms

Let $CI = \{x_1, x_2, ..., x_h\}$ be a set of closed itemset patterns, $CS = \{y_1, y_2, ..., y_h\}$ be a set of closed sequential patterns.

In [6], combining closed sequential pattern mining with closed itemset pattern mining can be performed in two following approaches.

- In the first approach, *CI* is firstly mined from multidimensional information using closed itemset pattern mining. Then *CCMD* (the set of candidate closed multidimensional sequential patterns) for each closed itemset pattern in *CI* is obtained, as shown below, by mining *CS* from sequences using closed sequential pattern mining. The union of all *CCMDs* is called $CCMD_{CI}$.

$$CCMD_1 = x_1 \times CS_1$$
$$CCMD_2 = x_2 \times CS_2$$
$$...$$
$$CCMD_h = x_h \times CS_h$$

- In the second approach, *CS* is firstly mined from sequences using closed sequential pattern mining. Then *CCMD* for each closed sequential pattern in *CS* is obtained, as shown below, by mining *CI* from multidimensional information using closed itemset pattern mining. The union of all *CCMDs* is called $CCMD_{CS}$.

$$CCMD_1 = y_1 \times CI_1$$
$$CCMD_2 = y_2 \times CI_2$$
$$...$$
$$CCMD_k = y_k \times CI_k$$

Analysis. From the first combination, it has to mine all sequences that occur with mined multidimensional information. Closed sequential pattern mining method must be called many times. It leads to a large cost of mining. Suppose *m* is the cost of closed itemset pattern mining algorithm, and *n* is the cost of closed sequential pattern mining algorithm. The first combination first mines multidimensional information in the whole dataset using closed itemset pattern mining algorithm, and then mines sequences that occur with mined multidimensional information using closed sequential pattern mining algorithm. Therefore, the running time is $O\ (m+(p.n))$, where p is the number of all closed itemset patterns. Moreover, this method leads to a large number of database scans being required. Because all sequences occurring with mined multidimensional information are scanned. Since the first combination computes on the closed itemset pattern mining algorithm and the closed sequential pattern mining algorithm, the total memory requirement is $O(t.m+(p.n))$, where t is number of transactions. The total database size is $t.\ |MD|$, where *MD* is set of multidimensional sequences. The fraction that fits in memory is given as $\alpha\ .t.|MD|$, where α is the fraction of database that fits in the memory. Therefore, the first combination takes the number of database scans as $(t.m+(p.n)) / (\alpha\ .t.|MD|) = (m+(p.n)) / (\alpha\ .|MD|)$.

There are similar problems with the second combination, it has to call closed itemset pattern mining algorithm many times to mine all multidimensional information that occurs with mined sequences. To reduce these problems, we know any pattern found in dataset *X* must also appear in dataset *Y* if $X \subseteq Y$ [2] as shown in lemma 1. Therefore, the new combination methods unnecessarily call closed sequential pattern mining method many times in the first combination, and don't need to call closed itemset pattern mining method many times in the second combination.

Lemma 1. If $X \subseteq Y$, any patterns (the set of closed sequential patterns and the set of closed itemset patterns) found in dataset *X* must also appear in dataset *Y*.

3.2. Proposed algorithms

Input: *SDB* and *min_support*.
Output: Set of candidate CMDS patterns
Method:
1. Mine multidimensional information in *SDB* by using closed itemset pattern mining.
2. For each closed itemset pattern in $A = \{a_1, a_2, a_3, ..., a_i\}$
   a. If (level =1)
      Mine a set of closed sequential patterns $CS_{level-1}$ from sequences associated with mined closed itemset pattern by using closed sequential pattern mining.
   b. Else
      call alongPattern($CS_{level-1}, a_{level-1}, a_i$)

Figure 1. CIScombine Algorithm

Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)
0-7695-2874-0/07 $25.00 © 2007 IEEE

Input: *SDB* and *min_support*.
Output: Set of candidate CMDS patterns
Method:
1. Mine sequences in *SDB* by using closed sequential pattern mining.
2. For each closed sequential patterns $S=\{s_1, s_2, s_3, \ldots, s_j\}$
   a. If (level=1)
      Mine a set of closed itemset patterns $CI_{level-1}$ from multidimensional information associated with mined closed sequential pattern by using closed itemset pattern mining.
   b. Else
      Call alongPattern($CI_{level-1}, s_{level-1}, s_j$)

Figure 2. CSIcombine Algorithm

Subroutine: alongPattern(P, m_1, m_l)
Parameters: P = a set of patterns(closed itemset patterns or closed sequential patterns), m_1 = a mined pattern at level 1, m_l = a mined pattern at level l where $l>1$
Return: A set of closed patterns CP associated with m_l
Method:
1. A set of different transaction identifiers t_{m1-ml}
2. If $\{p \in P \mid supp(p) - supp(p_{m_1-m_l}) \geq min\_support \wedge (\exists p' \in P \mid p \subset p' \wedge supp(p) = supp(p'))\}$
   a. $CP = p$

Figure 3. alongPattern subroutine

Figure 1-3 show two proposed algorithms, CIScombine and CSIcombine.

Example 5. Using CIScombine with table 1, all closed itemset patterns along with their closed sequential patterns are shown in the following patterns.

(6):3 <be>:2, <aba>:2, <abb>:2, <bcb>:2, <(bf)bf>:2
(8):3 <ba>:3, <bb>:3, <aba>:2, <abb>:2, <cba>:2, <(bd)ba>:2, <(bd)bc>:2, <bcb>:2
(6, 8):2 <aba>:2, <abb>:2
(6, 2):2 <bcb>:2, <be>:2, <cb>:2
(8, 1):2 <ba>:2, <bb>:2

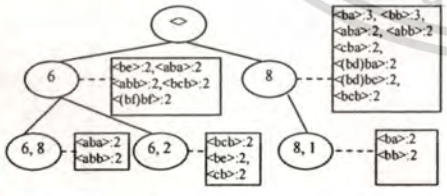


Figure 4. Closed patterns with their closed sequential patterns derived from table 1

Figure 4 shows all closed itemset patterns along with their closed sequential patterns. It shows that closed sequential patterns associated with lower-level closed itemset patterns can be found in the set of closed sequential pattern associated with first-level closed itemset patterns. Therefore, only sequences associated with first-level closed itemset patterns (minimal closed itemset patterns) are mined. It is unnecessary to mine sequences at lower-levels of the branch.

Theorem 1 (Computation cost). The running time of CIScombine is $O(m+(f.n)+(l.1))$, where m is the cost of closed itemset pattern mining, n is the cost of closed sequential pattern mining, f is the number of closed itemset patterns at the first level, and l is the number of closed itemset patterns at the lower level.

Proof. CIScombine first mines closed itemset patterns taking cost m. And then mines only closed sequential patterns along with closed itemset patterns at the first level. Each of the mining uses cost n. If there are f closed itemset patterns at the first level, this process takes $(f.n)$. At the lower level, there is no need to mine closed sequential patterns. Thus, this process takes $(l.1)$. The total running time of CIScombine is thus $O(m+(f.n)+(l.1))$.

Theorem 2 (Computation cost). The running time of CSIcombine is $O(n+(f.m)+(l.1))$, where m is the cost of closed itemset pattern mining, n is the cost of closed sequential pattern mining, f is the number of closed sequential patterns at the first level, and l is the number of closed sequential patterns at the lower level.

Proof. CSIcombine first mines closed sequential patterns taking cost n. And then mines only closed itemset patterns along with closed itemset patterns at the first level. Each of the mining uses cost m. There are f closed sequential patterns, so this process takes $(f.m)$. At the lower level, there is no need to mine closed sequential patterns. Thus, this process takes $(l.1)$. The total running time of CSIcombine is $O(n+(f.m)+(l.1))$.

Theorem 3 (I/O cost). The number of database scans made by CIScombine is given as $(m+(f.n)+(l.1)) / (\alpha .|MD|)$.

Proof. The number of database scans required is the total memory consumption of the algorithm divided by the fraction of database that will fit in memory. Since CIScombine computes on the closed itemset pattern mining and closed sequential pattern mining, the total memory requirement is $O(t.(m+(f.n)+(l.1)))$, where t is number of transactions. The total database size is $t.|MD|$, and the fraction that fits in memory is given as $\alpha .t.|MD|$; where α is the fraction of database that fits in memory and MD is set of multidimensional sequences. The number of database scans is given as

Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)
0-7695-2874-0/07 $25.00 © 2007 IEEE

$(t.(m + (f.n) + (l.1)) / (\alpha . t.|MD|) = (m + (f.n) + (l.1)) / (\alpha . |MD|)$.

Theorem 4 (I/O cost). The number of database scans made by CSIcombine is given as $(n+(f.m)+(l.1)) / (\alpha . |MD|)$.

Proof. The number of database scans required is the total memory consumption of the algorithm divided by the fraction of database that will fit in memory. Since CSIcombine computes on the closed itemset pattern mining and closed sequential pattern mining, the total memory requirement is $O(t.(n+(f.m)+(l.1)))$, where t is number of transactions. The total database size is $t.|MD|$, and the fraction that fits in memory is given as $\alpha . t.|MD|$; where α is the fraction of database that fits in memory and MD is set of multidimensional sequences. The number of database scans is given as $(t.(n + (f.m) + (l.1)) / (\alpha . t.|MD|) = (n + (f.m) + (l.1)) / (\alpha . |MD|)$.

3.2.1 Correctness

Theorem 5. A set of closed sequential patterns associated with a closed itemset pattern I (CS_I) is a subset of a set of closed sequential patterns associated with a closed itemset pattern I' ($CS_{I'}$) if $I \supset I'$.

Proof. Let $I \supset I'$. $I \supset I'$ means dataset containing I is a subset of dataset containing I'. So closed sequential patterns associated with I can be found in closed sequential patterns associated with I' according to lemma 1. Therefore, $CS_I \subset CS_{I'}$. □

Theorem 6. A set of closed itemset patterns associated with a closed sequential pattern s (CI_s) is a subset of a set of closed itemset patterns associated with a closed sequential pattern s' ($CI_{s'}$) if $s \supset s'$.

Proof. Let $s \supset s'$. $s \supset s'$ means dataset containing s is a subset of dataset containing s'. So closed itemset patterns associated with s can be found in closed itemset patterns associated with s' according lemma 1. Therefore, $CI_s \subset CI_{s'}$. □

Theorem 5 shows that we can derive closed sequential patterns associated with I from closed sequential patterns associated with I' in the first combination if we know that I is a superset of I'. We can take the same way in the second combination as shown in theorem 6.

4. Conclusion

We presented the new algorithms for mining multidimensional sequential patterns. The algorithms unnecessarily mine all datasets. Since any patterns (closed itemset patterns or closed sequential patterns) found in dataset X must also appear in dataset Y if $X \subseteq Y$. We can say that the closed sequential patterns associated with a closed itemset pattern I can be found from the closed sequential patterns associated with a sub-pattern of I when mining closed itemset pattern followed by mining closed sequential pattern mining. Thus, only sequences associated with first-level closed itemset patterns are mined. Sequences associated with lower-level closed itemset patterns are unnecessarily mined. We can take the same way when mining closed sequential patterns followed by mining closed itemset patterns. These algorithms take less the computational cost than existing algorithms. In addition, they avoid database scans in some passes which leads to reduce disk I/O cost.

However, these algorithms do not outperform existing algorithms without existing of super-set patterns. Although these algorithms can reduce computation time and the number of database scans, the redundant patterns still are generated from both combinations.

5. References

[1] C. Yu and Y. Chen, "Mining Sequential Patterns from Multidimensional sequence data", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 1, January 2005, pp. 136-140.

[2] H. Pinto, Multi-dimensional Sequential Pattern Mining, M. Sc. Thesis, Simon Fraser University, Canada, April 2001.

[3] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen and U. Dayal, "Multi-dimensional Sequential Pattern Mining", In Proc. 2001 ACM CIKM Int. Conf. Information and Knowledge Management(ACM'01), Atlanta, Georgia, USA, November 5-10, 2001, pp. 81-88.

[4] M. Plantevit, Y. W. Choong, A. Laurent, D. Laurent and M. Teisseire, "M2SP:Mining Sequential Patterns among Several Dimensions", *Knowledge Discovery in Database PKDD*, Vol 3721, 2005, pp. 205-216.

[5] M. Plantevit, Y. W. Choong, A. Laurent, D. Laurent and M. Teisseire, "Mining Closed Multidimensional Sequential Patterns", document report, 2006.

[6] P. Songram, V. Boonjing and S. Intakosum, "Closed multidimensional sequential pattern mining", In Proc. of the 3rd IEEE Int. Conf. on Information Technology : New Generations (ITNG'06), Las Vegas, Nevada, USA, 10-12 April, 2006, pp.512-517.

Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)
0-7695-2874-0/07 $25.00 © 2007 IEEE

BIOGRAPHY

Panida Songram was born on April 25, 1978 in Burirum, Thailand. She earned her bachelor's degree with honour 2 in computer scicence from Mahasarakham University in 2001. She then graduated her master's degree in computer science from King Mongkut's institute of Technology Ladkrabang in 2004, and got a scholarship from Ministry of Science and Technology for training an internet programming course in India. She then continuted her Ph.D in computer scicne in King Mongkut's institute of Technology Ladkrabang. During her Ph.D study, she got a scholarship from Higher Education Commission for doing her research at Taiwan Ocean University, Taiwan. In addition, she presented her research at China, Austalia and Korea.