# Multi-dimensional Sequential Pattern Mining*

Helen Pinto Jiawei Han Jian Pei Ke Wang
Intelligent Database Systems Research Lab. School of Computing Science
Simon Fraser University, Burnaby, B.C., Canada V5A 1S6
E-mail: {hlpinto, han, peijian, wangk}@cs.sfu.ca

Qiming Chen Umeshwar Dayal
Hewlett-Packard Labs.
1501 Page Mills Road, P.O. Box 10490, Palo Alto, California 94303-0969, U.S.A.
E-mail: {qchen, dayal}@hpl.hp.com

## ABSTRACT

Sequential pattern mining, which finds the set of frequent subsequences in sequence databases, is an important data-mining task and has broad applications. Usually, sequence patterns are associated with different circumstances, and such circumstances form a multiple dimensional space. For example, customer purchase sequences are associated with region, time, customer group, and others. It is interesting and useful to mine sequential patterns associated with multi-dimensional information.

In this paper, we propose the theme of multi-dimensional sequential pattern mining, which integrates the multidimensional analysis and sequential data mining. We also thoroughly explore efficient methods for multi-dimensional sequential pattern mining. We examine feasible combinations of efficient sequential pattern mining and multi-dimensional analysis methods, as well as develop uniform methods for high-performance mining. Extensive experiments show the advantages as well as limitations of these methods. Some recommendations on selecting proper method with respect to data set properties are drawn.

## 1. INTRODUCTION

Sequential pattern mining [1], i.e., mining frequent subsequences as patterns in a sequence database, is an important data mining task with broad applications, including the analysis of customer behaviors, Web access patterns, process analysis of scientific experiments, prediction of natural disasters, disease treatments, drug testing, DNA analysis, etc.

As an example, an ISP (Internet Service Provider) may find from its customer purchase database a sequential pattern $P_1$ = *"try a* 100 *hour free internet access package"* → *"subscribe to* 15 *hours/month package"* → *"upgrade to* 30 *hours/month package"* → *"upgrade to unlimited package"* holds for 32% of customers. Such patterns can be used to develop marketing and product strategies.

The patterns found from sequential pattern mining, though uncover global regularity among customers, may suffer from a lack of focus. For example, the above sequential pattern $P_1$ may not be popular for customers over 55. Many of these older customers may use their access package only to check email every two or three days and hence the 30 hours/month package is their preferred choice. Thus, sequential pattern $P_2$ = *"try* 100 *hour free package* → *subscribe to* 30 *hours/month package"* may hold for customers over 55. On the other hand, pattern $P_1$ may hold for a much higher percentage, say 75%, of professional customers younger than 35.

Clearly, if sequential pattern mining can be associated with customer cateogry or other multi-dimensional information, it will be more effective since the classified patterns are often more useful. Simular situations exist in many practical applications. This motivates our study of *multi-dimentional sequential pattern mining.*

Recent studies highlighted *multi-dimentional analysis* as another frontier of data mining research. For example, frequent patterns can be associated with transactions happening at different circumstances [5], which forms a tyipcal case of multi-dimensional association mining. However, there is no previous study on mining sequential patterns in multi-dimensional circumstances.

In this paper, we integrate sequential pattern mining and multi-dimensional analysis and propose the theme of *multi-dimensional sequential pattern mining.* Several possible efficient methods are proposed for multi-dimensional sequential pattern mining. This can be categorized into two categories: (1) integration of efficient sequential pat-

*The work was supported in part by the Natural Sciences and Engineering Research Council of Canada (grant NSERC-A3723), the Networks of Centres of Excellence of Canada (grant NCE/IRIS-3), and the Hewlett-Packard Lab, U.S.A.

tern mining and multi-dimensional analysis methods, and (2) embedding multi-dimensional information into sequences and mine the whole set using a uniform sequential pattern mining method. Our extensive experiments show the advantages as well as limitations of these methods. Some recommendations on selecting proper methods with respect to data set properties are drawn.

The remaining of the paper is organized as follows. In section 2, we define the problem of multi-dimensional sequential pattern mining and revisit related work. Section 3 introduces *UniSeq*, an algorithm by embedding multi-dimensional information into sequences. Section 4 develops two algorithms, *Seq-Dim* and *Dim-Seq*, which integrate sequential pattern mining and multi-dimensional frequent pattern mining in two different ways. An extensive performance study comparing the three methods is reported in Section 5. We discuss related issues and potential extensions and conclude the paper in Section 6.

## 2. PROBLEM DEFINITION

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of **items**. An **itemset** $X$ is a subset of items, i.e., $X \subseteq I$. A **sequence** is an ordered list of itemsets. A sequence $s$ is denoted by $\langle s_1 s_2 \cdots s_l \rangle$, where $s_j$ is an itemset, i.e., $s_j \subseteq I$ for $1 \leq j \leq l$. $s_j$ is also called an **element** of the sequence, and denoted as $(x_1 x_2 \cdots x_m)$, where $x_k$ is an item, i.e., $x_k \in I$ for $1 \leq k \leq m$. For brevity, the brackets are omitted if an element has only one item. That is, element $(x)$ is written as $x$. An item can occur at most once in an element of a sequence, but can occur multiple times in different elements of a sequence. The number of instances of items in a sequence is called the **length** of the sequence. A sequence with length $l$ is called an **$l$-sequence**. A sequence $\alpha = \langle a_1 a_2 \cdots a_n \rangle$ is called a **subsequence** of another sequence $\beta = \langle b_1 b_2 \cdots b_m \rangle$ and $\beta$ a **super sequence** of $\alpha$, denoted as $\alpha \sqsubseteq \beta$, if there exist integers $1 \leq j_1 < j_2 < \cdots < j_n \leq m$ such that $a_1 \subseteq b_{j_1}$, $a_2 \subseteq b_{j_2}$, $\ldots$, $a_n \subseteq b_{j_n}$.

A **sequence database** $S$ is a set of tuples $\langle sid, s \rangle$, where $sid$ is an **identification** of the sequence and $s$ a sequence. A tuple $\langle sid, s \rangle$ is said to *contain* a sequence $\alpha$, if $\alpha$ is a subsequence of $s$, i.e., $\alpha \sqsubseteq s$. The support of a sequence $\alpha$ in a sequence database $S$ is the number of tuples in the database containing $\alpha$, i.e., $support_S(\alpha) = |\ \{\langle sid, s \rangle | (\langle sid, s \rangle \in S) \wedge (\alpha \sqsubseteq s)\}\ |$. It can be denoted as $support(\alpha)$ if the sequence database is clear from the context. Given a positive integer $min\_support$ as the **support threshold**, a sequence $\alpha$ is called a **sequential pattern** in sequence database $S$ if the sequence is contained by at least $min\_support$ tuples in the database, i.e., $support_S(\alpha) \geq min\_support$. A sequential pattern with length $l$ is called an **$l$-pattern**.

EXAMPLE 1. *Let our running database be SDB given in Table 1. The database records the attributes and purchase history of customers. There are three dimensions, customer-group (cust-grp), city and age-group (age-grp). Customers are identified by* customer-id *(cid). Let a, b, ..., h be items bought by customers. The customer-ids and purchase history (the first and last columns in the table) form a* sequence database, *where cid is used for sequence identification. Suppose the* support threshold *$min\_support = 2$.*

| cid | cust-grp | city | age-grp | sequence |
|---|---|---|---|---|
| 10 | business | Boston | middle | $\langle (bd)cba \rangle$ |
| 20 | professional | Chicago | young | $\langle (bf)(ce)(fg) \rangle$ |
| 30 | business | Chicago | middle | $\langle (ah)abf \rangle$ |
| 40 | education | New York | retired | $\langle (be)(ce) \rangle$ |

**Table 1: A multi-dimensional sequence database**

*A* sequence *$\langle (bd)cba \rangle$ has 4* elements*: $(bd)$, $c$, $b$ and $a$. It is a* 5-sequence *since there are 5 instances appearing in that sequence. Sequence $\langle bc \rangle$ is a* subsequence *of $\langle (bd)cba \rangle$. Sequence $\langle bc \rangle$ is also a* sequential pattern *since it is contained in tuple* 10, 20 *and* 40. *Thus, $support(\langle bc \rangle) = 3$. Its support passes support threshold.* □

A **multi-dimensional sequence database** is of schema $(RID, A_1, \ldots, A_m, S)$, where $RID$ is a primary key, $A_1, \ldots, A_m$ are dimensions and $S$ is in the domain of sequences. Let $*$ be a meta-symbol which does not belong to any domain of $A_1, \ldots, A_m$. A **multi-dimensional sequence** takes the form of $(a_1, \ldots, a_m, s)$, where $a_i \in (A_i \cup \{*\})$ for $(1 \leq i \leq m)$ and $s$ is a sequence. A multi-dimensional sequence $P = (a_1, \ldots, a_m, s)$ is said to **match** a tuple $t = (x_1, \ldots, x_m, s_t)$ in the multi-dimensional sequence database if and only if, for $(1 \leq i \leq m)$, either $a_i = x_i$ or $a_i = *$, and $s \sqsubseteq s_t$. The number of tuples in the database matching multi-dimensional sequence $P$ is called the **support** of $P$, denoted as $support(P)$. Given a minimum support threshold $min\_support$, a multi-dimensional sequence $P$ is called a **multi-dimensional sequential pattern** if and only if $support(P) \geq min\_support$.

EXAMPLE 2. *Together, all the columns in Table 1 form a multi-dimensional sequence database. A multi-dimensional sequence $P = (business, *, *, \langle b \rangle)$ matches tuple $(10, business, Boston, middle, \langle (bd)cba \rangle)$. The support of P in SDB is 2. Therefore, P is a* multi-dimensional sequential pattern. □

Many studies have contributed to the efficient mining of sequential patterns or other frequent patterns in time-related data [1, 14, 9, 16, 17, 10, 8, 2, 11, 13, 6]. Srikant and Agrawal [14] generalize their definition of sequential patterns in [1] to include time constraints, sliding time window, and user-defined taxonomy. Mannila, et al. [9] present a problem of mining frequent episodes in a sequence of events, where episodes are essentially acyclic graphs of events whose edges specify the temporal before-and-after relationalship but without timing-interval restrictions. Bettini, et al. [2] consider a generalization of inter-transaction association rules. These are essentially rules whose left-hand and right-hand sides are episodes with time-interval restrictions. Lu, et al. [8] propose inter-transaction association rules which are implication rules whose two sides are totally-ordered episodes with timing-interval restrictions. Garofalakis, et al. [4] propose the use of regular expressions as a flexible constraint specification tool that enables user-controlled focus to be incorporated into the sequential pattern mining process.

Almost all of the proposed methods for mining sequential patterns are based on the *Apriori* heuristic. The heuristic states the fact that *any super-pattern of an infrequent pattern cannot be frequent.*

Based on this heuristic, a typical *Apriori*-like method, such as *GSP* [14], adopts a multiple-pass, candidate generation-and-test approach. The first scan finds all of the frequent items which form the set of single item frequent sequences. Each subsequent pass starts with a *seed set* of sequential patterns, which is *the set of sequential patterns found in the previous pass.* This seed set is used to generate new potential patterns, called *candidate sequences.* Each candidate sequence contains one more item than a seed sequential pattern, where each element in the pattern may contain one item or multiple items. So, all the candidate sequences in a pass will have the same length. The scan of the database in one pass finds the support for each candidate sequence. All the candidates whose support in the database is no less than *min_support* form the set of the newly found sequential patterns. This set then becomes the seed set for the next pass. The algorithm terminates when no new sequential pattern is found in a pass, or when no candidate sequence can be generated.

The *Apriori*-like sequential pattern mining methods, though reduce the search space, bear two major and inherent costs independent of the implementation techniques. First, they may have to generate a very large set of candidate sequences. Second, they may have to scan database many times when long patterns exist. To overcome these problems, an efficient sequential pattern mining method, *PrefixSpan,* is developed in [12]. In this study, we use *PrefixSpan* as the sequential pattern mining method. The idea of *PrefixSpan* will be illustrated in our example in Section 3.

## 3. UNISEQ: EMBED MULTIDIMENSIONAL INFORMATION INTO SEQUENCES

Conceptually, for a tuple $t$ in a multi-dimensional sequence database, the multi-dimensional information can be embedded in the sequence by introducing a special element. For example, for tuple $t = (10,$ *business*, *Boston*, *middle*, $\langle(bd)cba\rangle)$ in Table 1, sequence $s = \langle(bd)dba\rangle$ in $t$ can be extended to $s^{md} = \langle(business\ Boston\ middle)(bd)cba\rangle$. Then, mining multi-dimensional sequential patterns in a multi-dimensional sequence database can be done by mining the extended sequence database. The mining process is demonstrated in the following example.

EXAMPLE 3. *Let us consider mining multi-dimensional sequential patterns from database $SDB$ in Table 1.*

*By extending an element at the beginning of every sequence in the database, we embed multi-dimensional information and get a sequence database $SDB^{MD}$, as shown in Table 2. Such an extension of sequence is called an* **MD-extension***, where multi-dimensional information is put as the first element of the sequence.*

| cid | MD-extension of sequence |
|---|---|
| 10 | $\langle(business\ Boston\ middle)(bd)cba\rangle$ |
| 20 | $\langle(professional\ Chicago\ young)(bf)(ce)(fg)\rangle$ |
| 30 | $\langle(business\ Chicago\ middle)(ah)abf\rangle$ |
| 40 | $\langle(education\ Atlanta\ retired)(be)(ce)\rangle$ |

**Table 2: MD-extension database $SDB^{MD}$ from the multi-dimensional sequence database $SDB$ shown in Table 1**

*Sequence database $SDB^{MD}$ can be mined using* PrefixSpan*, as follows.*

*In the first scan of the database,* PrefixSpan *finds all the single-item frequent sequences. These are $\langle business\rangle : 2$, $\langle Chicago\rangle : 2$, $\langle middle\rangle : 2$, $\langle a\rangle : 2$, $\langle b\rangle : 4$, $\langle c\rangle : 3$, $\langle e\rangle : 2$ and $\langle f\rangle : 2$. The complete set of sequential patterns can then be partitioned into 8 subsets, each with one of the single-item sequences as prefix. Each subset is mined by constructing its corresponding* projected database *and recursively mining it. A* **projected database** *consists of* postfix *sequences, and a* **postfix sequence** *contains all those frequent items that follow the first occurrence of a given prefix in any sequence. In cases where the first postfix item is in the same element as the last prefix item, it is indicated as (_item).*

*For example, the $\langle Chicago\rangle$-projected database contains two* postfix *sequences: $\langle(bf)(ce)f\rangle$ and $\langle middle\ aabf\rangle$. First, we print out the sequential pattern $\langle Chicago\rangle$, then find the single-item frequent sequences in this projected database. They are: $\langle b\rangle$ and $\langle f\rangle$, which form the sequential patterns "$\langle Chicago\ b\rangle : 2$" and "$\langle Chicago\ f\rangle : 2$" respectively. Projecting each of these 2-item prefixes further, we see that the $\langle Chicago\ f\rangle$-projected database does not contain enough sequences for any item within it to satisfy min_support. However, $\langle Chicago\ b\rangle$-projected database contains postfix sequences: $\langle(\_f)f\rangle$ and $\langle f\rangle$ with one frequent item between them, i.e., $f$. This yields the sequential pattern "$\langle Chicago\ bf\rangle : 2$." Since the $\langle Chicago\ bf\rangle$-projected database does not satisfy min_support, the processing of the subset prefixed by $\langle b\rangle$ stops.*

*The projected databases for length-1 patterns as well as the patterns mined from them are shown in Table 3.*

*Every sequential pattern mined in the above process corresponds to a multi-dimensional sequential pattern. For example, pattern $\langle Chicago\ bf\rangle$ corresponds to multidimensional sequential pattern $(*, Chicago, *, \langle bf\rangle)$, which represents that customers in Chicago who purchase item $b$ followed by $f$ in a later transaction.* □

Now, let us verify the multi-dimensional sequential pattern mining using MD-extension database.

THEOREM 3.1 (*UniSeq*). *Let $SDB$ be a multidimensional sequence database and $SDB^{MD}$ be the MD-extension of $SDB$. A multi-dimensional sequence $(a_1, \ldots, a_n, s)$ is*

| Prefix | Projected (postfix) database | Sequential patterns |
|---|---|---|
| $\langle business\rangle$ | $\langle middle\ bcda\rangle$, $\langle middle\ aabf\rangle$ | $\langle business\rangle$, $\langle(business\ middle)\rangle$, $\langle(business\ middle)a\rangle$, $\langle(business\ middle)b\rangle$, $\langle business\ a\rangle$, $\langle business\ b\rangle$ |
| $\langle Chicago\rangle$ | $\langle(bf)(ce)f\rangle$, $\langle middle\ aabf\rangle$ | $\langle Chicago\rangle$, $\langle Chicago\ b\rangle$, $\langle Chicago\ f\rangle$, $\langle Chicago\ bf\rangle$ |
| $\langle middle\rangle$ | $\langle bcba\rangle$, $\langle aabf\rangle$ | $\langle middle\rangle$, $\langle middle\ a\rangle$, $\langle middle\ b\rangle$ |
| $\langle a\rangle$ | $\langle abf\rangle$ | $\langle a\rangle$ |
| $\langle b\rangle$ | $\langle cba\rangle$, $\langle(\_f)(ce)f\rangle$, $\langle f\rangle$, $\langle(\_e)(ce)\rangle$ | $\langle b\rangle$, $\langle bc\rangle$, $\langle b(ce)\rangle$, $\langle be\rangle$, $\langle bf\rangle$ |
| $\langle c\rangle$ | $\langle ba\rangle$, $\langle(\_e)bf\rangle$, $\langle(\_e)\rangle$ | $\langle c\rangle$, $\langle(ce)\rangle$ |
| $\langle e\rangle$ | $\langle f\rangle$ | $\langle e\rangle$ |
| $\langle f\rangle$ | | $\langle f\rangle$ |

**Table 3: Projected databases and sequential patterns obtained via *UniSeq***

*a multi-dimensional sequential pattern in $SDB$ if and only if sequence $s^{md} = \langle(a_1 \cdots a_n)s\rangle$ is a sequential pattern in $SDB^{MD}$.*

**Proof.** *If $\langle(a_1 \cdots a_n)s\rangle$ is not a sequential pattern in $SDB^{MD}$, then values $a_1, \ldots, a_n$ do not all occur frequently with $s$ and hence $(a_1, \ldots, a_n, s)$ cannot be a frequent multi-dimensional sequence in $SDB$. If $(a_1, \ldots, a_n, s)$ is a multi-dimensional sequence in $SDB$, then $a_1, \ldots, a_n$ occur frequently with $s$ and can be arbitrarily ordered within the first element of $s^{md}$ to give pattern $\langle(a_1 \cdots a_n)s\rangle$ which must also be frequent.* □

Based on Theorem 3.1 and Example 3, we have the multi-dimensional sequential pattern mining algorithm using MD-extension database as follows.

Algorithm 1 (*UniSeq* ).

**Input:** *Multi-dimensional sequence database $SDB$ and support threshold min_sup.*

**Output:** *The complete set of multi-dimensional sequential patterns.*

**Method:** *Let $SDB^{MD}$ be the MD-extension database of $SDB$. Mine sequential patterns in $SDB^{MD}$ using* PrefixSpan*. For each sequential pattern $P$ in $SDB^{MD}$, output the corresponding multi-dimensional sequential pattern in $SDB$.* □

**Rationale.** *The correctness and completeness of the algorithm follow Theorem 3.1 immediately.* □

As an alternative, instead of embedding the multidimensional information into the first element of each sequence, it can be attached as the last element. For example, by concatenating an element at the end of every sequence in the database, we embed multi-dimensional information and get a sequence database $SDB'$, as shown in Table 4.

Sequence database $SDB'$ can be mined using *PrefixSpan*. It is easy to see that every sequential pattern in $SDB'$ corresponds to a multi-dimensional sequential pattern in $SDB$. For example, $\langle bf\ Chicago\rangle$ corresponds to $(*, Chicago, *, \langle bf\rangle)$.

Although multi-dimensional information can be embedded into either the first or the last elements of each sequence, our experiments show that the two alternatives have almost identical performance results. We call this method *UniSeq* (*uniform sequential*).

| cid | Extension of sequence by attaching an element |
|---|---|
| 1 | $\langle(bd)cba(business\ Boston\ middle)\rangle$ |
| 2 | $\langle(bf)(ce)(fg)(professional\ Chicago\ young)\rangle$ |
| 3 | $\langle(ah)abf(business\ Chicago\ middle)\rangle$ |
| 4 | $\langle(be)(ce)(education\ Atlanta\ retired)\rangle$ |

**Table 4: Extension database $SDB'$ from a multi-dimensional sequence database $SDB$ in Table 1.**

*UniSeq* mines multi-dimensional sequential patterns by embedding multi-dimensional information into each sequence and then applying *PrefixSpan* to the extended sequence database. The advantage of *UniSeq* is that it reduces the problem to mining one extended sequence database, and is therefore easy to implement. Since all dimension values are treated as sequential items, the drawback of this method is that it cannot take advantage of efficient mining algorithms of multi-dimensional nonsequential computation methods, such as BUC [3] or H-cubing [7], and thus leads to less efficient computation when the cost of computing multi-dimensional values becomes substantial (e.g., when the number of dimensions is not very small).

## 4. COMBINE ICEBERG CUBING AND SEQUENTIAL PATTERN MINING

Given a multi-dimensional sequence database $SDB$ with schema $(RID, A_1, \ldots, A_m, s)$, the information in each tuple $t = (rid, x_1, \ldots, x_m, s_t)$ can be partitioned into two parts: *dimensional information* $(x_1, \ldots, x_m)$ and *sequence* $s_t$. It is then natural to divide the mining process into two steps: first mine patterns about dimensional information, and then find sequential patterns from projected sub-database, or vice versa. The observations are shown in the following example.

Example 4. *Let us re-examine the multi-dimensional sequential patterns in database $SDB$ (Table 1). The support threshold is set to 2.*

*One can* first find frequent multi-dimensional value combinations, then find correponding sequential patterns. *For example, since $(*, Chicago, *)$ is contained in tuples* 20 *and* 30*, it is frequent. Such frequent multi-dimensional value combinations are called* **multi-dimensional patterns***, or* **MD-patterns***.*

*Then, all the sequences in tuples containing MD-pattern* $P = (*, Chicago, *)$ *are collected. They form the* **multi-dimensional pattern projected database***, or* **MD-projected database** *for P, denoted as* $SDB|_P$*. There are two sequences in* $SDB|_P$*:* $\langle(bf)(ce)(fg)\rangle$ *and* $\langle(ah)abf\rangle$*. Mine sequential patterns within* $SDB|_P$*. For example,* $\langle bf\rangle$ *is a sequential pattern in* $SDB|_P$*, thus* $(*, Chicago, *, \langle bf\rangle)$ *is a multi-dimensional sequential pattern.*

*Alternatively, one can* first find sequential patterns, and then find their corresponding MD-patterns*. For example, by mining sequence database consisting of the first and fourth columns in Table 1, sequential pattern* $s = \langle bf\rangle$ *is identified. Then, collect all multi-dimensional information in tuples containing s:* $(professional, Chicago, young)$ *and* $(business, Chicago, middle)$*. They form the* **projected multi-dimensional database***, or* **projected MD-database** *for s, denoted as* $SDB|_s$*.*

*Then, one can mine* $MD - patterns$ *in* $SDB|_s$*. For example,* $(*, Chicago, *)$ *is an MD-pattern. Therefore,* $(*, Chicago, *, \langle bf\rangle)$ *is a multi-dimensional sequential pattern.* □

In general, the soundness of the methods proposed in Example 4 can be verified by the following theorem.

THEOREM 4.1. *Given a multi-dimensional sequence* $P = (a_1, \ldots, a_m, s)$*, let* $P_d = (a_1, \ldots, a_m, \langle\rangle)$ *and* $P_s = (\underbrace{*, \ldots, *}_{m}, s)$*.*

1. *Let* $SDB|_{P_d}$ *be the set of tuples in SDB matching* $P_d$*. P is a multi-dimensional sequential pattern if and only if* $P_d$ *and* $P_s$ *are multi-dimensional sequential patterns in SDB and* $SDB|_{P_d}$*, respectively.*

2. *Let* $SDB|_{P_s}$ *be the set of tuples in SDB matching* $P_s$*. P is a multi-dimensional sequential pattern if and only if* $P_s$ *and* $P_d$ *are multi-dimensional sequential patterns in SDB and* $SDB|_{P_s}$*, respectively.*

**Proof.** *The theorem follows related definitions immediately.* □

Based on Theorem 4.1, the problem of multi-dimensional sequential pattern mining problem can be reduced to two sub-problems: sequential pattern mining and MD-pattern mining. As introduced before, sequential pattern mining can be done efficiently by *PrefixSpan*. For MD-pattern mining, we adopt a $BUC$-like algorithm, where $BUC$ is an efficient iceberg cube computing algorithm developed in [3]. The general idea of our $BUC$-like algorithm is illustrated in the following example.

EXAMPLE 5. *Let us consider finding MD-patterns, i.e., frequent multi-dimensional value combinations in the multi-dimensional database consisting of the second, third and fourth columns in Table 1. Let the support threshold be* 2*.*

1. *First, sort all tuples in the database in alphabetical order of values in dimension cust-grp. Since there is only one tuple having value education, any multi-dimensional value combination having education cannot be an MD-pattern. So, there is no need to search the customer group of education. The same principle applies to customer group professional. Group business contains two tuples. Therefore, an MD-pattern* $(business, *, *)$ *is found, and the group needs to be analyzed further.*

   (a) *Within group business, sort tuples in alphabetical order of values in dimension city. Since each city group has only one tuple, no MD-pattern with minimum support* 2 *can be formed. So, one can ignore dimension city in the analysis of customer group business.*

   (b) *Then, within group business, sort tuples in alphabetical order of values in dimension age-grp. A sub-group* $(business, *, middle)$ *contains two tuples. Thus,* $(business, *, middle)$ *is an MD-pattern. Since there is no more dimension at this point, the search returns.*

   *After analyzing dimension cust-grp, this dimension can be excluded from the remaining mining, since all MD-patterns having a non-"*" value in this dimension have been found.*

2. *Then, one can start analyzing dimension city. Similarly, sort tuples in alphabetical order of values in dimension city. Only group Chicago having* 2 *tuples passing support threshold. A pattern* $(*, Chicago, *)$ *is output and the group is analyzed recursively.*

3. *At last, by analyzing dimension age-grp, one can find MD-pattern* $(*, *, middle)$*.*

*In summary, the processing tree of the* BUC*-like algorithm is shown in Figure 1. The tree is expanded further if and only if a sub-group has enough tuples. The correctness of the algorithm is shown in this example and also verified in [3].* □

Based on Theorem 4.1, two algorithms are developed for multi-dimensional sequential pattern mining, as shown below.

ALGORITHM 2 (*Dim-Seq* AND *Seq-Dim*).

**Input and output:** *same as algorithm 1.*

**Method:**

***Dim-Seq*:** *First find MD-patterns. For each MD-pattern, form MD-projected database then mine sequential patterns in projected databases.*

***Seq-Dim*:** *First mine sequential patterns. For each sequential pattern, form projected MD-database and then find MD-patterns within projected databases.*

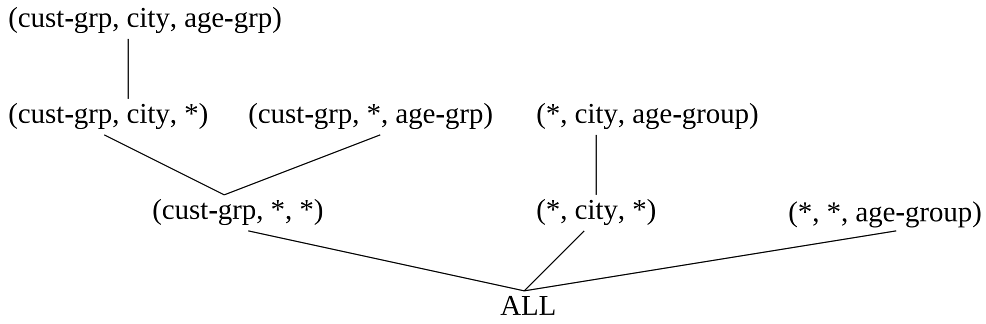

**Figure 1: *BUC* processing tree for $SDB$**

**Rationale.** *The correctness and completeness of algorithm* Dim-Seq *and* Seq-Dim *follow the first and second cases in Theorem 4.1.* □

Both algorithms *Dim-Seq* and *Seq-Dim* are correct and complete. However, *Seq-Dim* should be more efficient in general as shown below. In *Dim-Seq*, the mining of dimensions are shared in the multiple sequential database, whereas the mining of sequential patterns for different dimension combinations are separated. Different dimension combinations may share many common sequences, but the method cannot explore the shared mining of such sequences. In contrast, *Seq-Dim* mines one sequence database to derive all the sequential patterns. The saving of *Seq-Dim* from the mining of many small sequential databases as in *Dim-Seq* makes the method more efficient. Such an analysis is also supported by our performance study.

## 5. EXPERIMENTAL RESULTS AND PERFORMANCE STUDY

In this section, we report our experimental results on the performance of three algorithms: *UniSeq*, *Dim-Seq* and *Seq-Dim*. Our performance study shows that *Seq-Dim* is a scalable and efficient method. It outperforms the other two methods in many cases.

Our experiments were run on a 800 MHz Pentium III PC with 1 gigabyte main memory. All the methods are implemented using Microsoft Visual C++ 6.0. As mentioned in Section 3, there are two ways to implement *UniSeq*. Our experimental results show that both methods have almost identical performance. Thus, we only show the performance of the one which puts all the dimensional information in the last elements of each sequence.

We use synthetic datasets to test the three methods. In the synthetic datasets, sequences are generated using a standard procedure described in [15]. Extensive experiments were performed over many datasets. The results are consistent in trend. Limited by space, only a set of experiments over one data set is reported here. In this dataset, the number of items is set to 10,000, while the number of sequences is $10,000$. 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 randomly so that values are distributed evenly in every dimension.

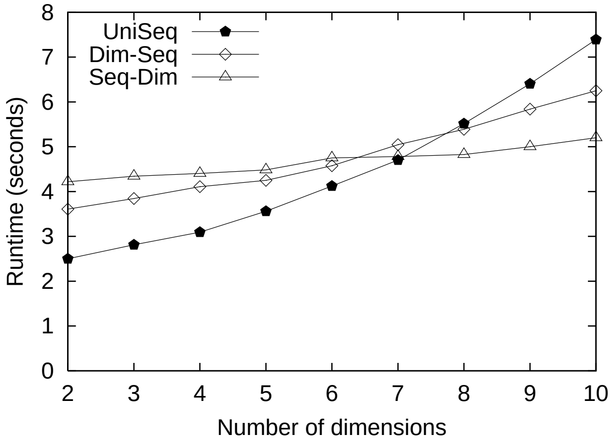

**Figure 2: Scalability over dimensionality.**

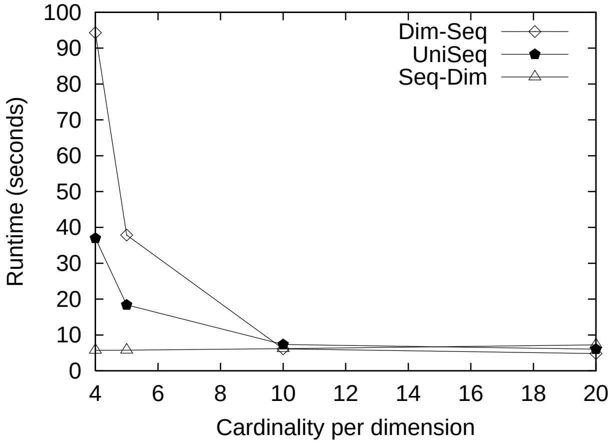

**Figure 3: Scalability over cardinality.**

Figure 2 shows the scalability of the algorithms over the number of dimensions. The support threshold is set to 0.25%. The cardinality of each dimension is set to 10. As the dimensionality increases, the runtimes of all the three algorithms go up. However, *Seq-Dim* is more scalable.

When dimensionality is low, the major cost is in mining sequential patterns. The elements containing multi-dimensional information are short and *PrefixSpan* can handle them easily. Thus, *UniSeq* outperforms the other two methods.

When dimensionality is high, the major cost is mining multi-dimensional information. *Seq-Dim* is faster because it only mines multi-dimensional patterns that occur with an existing sequential pattern. *Dim-Seq* is also faster than *UniSeq*. That is because *UniSeq* has to deal with longer sequences and patterns when many dimension values are included.

Figure 3 shows the scalability of the algorithms over cardinality. There are 10 dimensions and the support threshold is set to 0.25%. Various cardinalities are achieved by proper mapping of dimension values. When cardinality is high, the database becomes sparse. All methods have similar performance. However, when cardinality is low, the database becomes dense. Both *UniSeq* and *Dim-Seq* encounter the difficulty of dealing with many patterns. For example, before finding any sequential patterns, *Dim-Seq* must first explore all frequent multi-dimensional combinations, even though some of them may not lead to any multi-dimensional sequential pattern. *Seq-Dim* avoids those costs. It only explores multi-dimensional combination under the condition of some sequential patterns. That is the reason why it outperforms the others significantly.

Figure 4 shows the scalability of the algorithms over support threshold. Here, the dimensionality and cardinality are set to 8 and 10, respectively. It can be seen that all methods scale well.

Figure 5 shows the scalability of the algorithms over the number of sequences in the database. The database size ranges from $10,000$ to $20,000$, and the support threshold is set to 0.25%. The dimensionality and cardinality of each dimension are both set to 10. Both *Dim-Seq* and *UniSeq* scale linearly but *Seq-Dim* is better. As the database becomes larger, there could be many frequent multi-dimensional values which lead to no multi-dimensional sequential patterns. That makes *Dim-Seq* scale poorly.

In summary, the advantages and disadvantages of the algorithms are as follows.

- ***Seq-Dim* is efficient and scalable**. It is the fastest algorithm in most cases.

  Comparing with *Dim-Seq*, *Seq-Dim* first looks at sequential patterns. It explores MD-patterns only if there is some sequential pattern found. That makes the search more fruitful. Therefore, *Seq-Dim* is more scalable than *Dim-Seq*. In most cases, *Seq-Dim* is also more efficient than *Dim-Seq*.

  When mining dense datasets or datasets with high dimensionality, *Seq-Dim* has advantages over *UniSeq*. That is because the *BUC*-like method is more capable than *PrefixSpan* in finding multi-dimensional patterns in high dimensional space.

- ***UniSeq* is also an efficient and scalable method**. It is the fastest method among the three when dimensionality is low.

  When dimensionality is low, the advantage of *BUC*-like method on finding multi-dimensional patterns is minor. On the other hand, since *UniSeq* is fully based on *PrefixSpan*, it does not need any overhead on switching data structure and mining process. Thus *UniSeq* wins.

  The major cost of *UniSeq* is mining the elemets with multi-dimenisonal values. When dimensionality is high, such elements is long and thus *UniSeq* has to handle long sequences and patterns.

- **The scalability of *Dim-Seq* is not good comparing with the other two methods**.

  *Dim-Seq* has two major burdens: (1) before touching the sequences, it has to find multi-dimensional patterns. Many multi-dimensional patterns may not lead to multi-dimensional sequential patterns. Finding such multi-dimensional patterns is fruitless. (2)

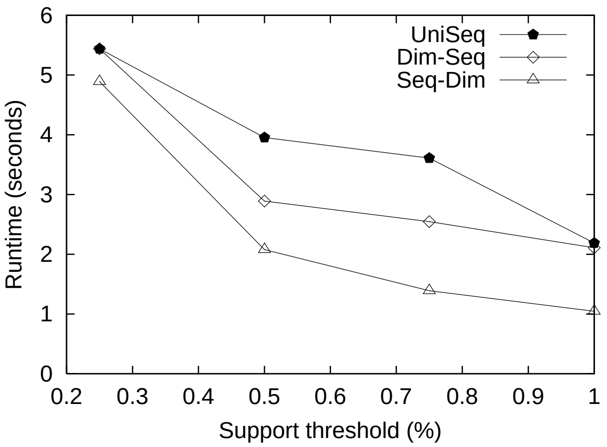

**Figure 4: Scalability over support threshold.**

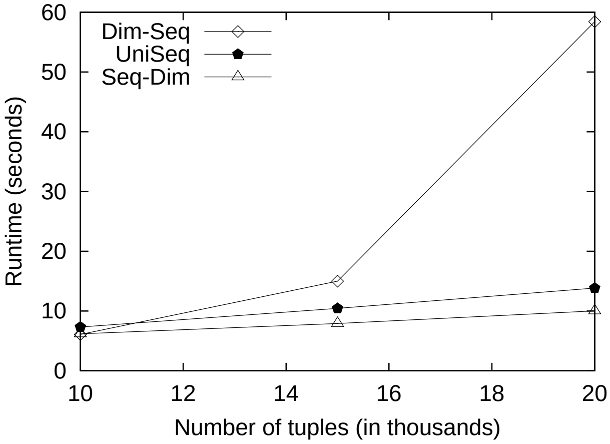

**Figure 5: Scalability over number of sequences.**

no optimization of mining sequential patterns can be applied to *Dim-Seq*. When dimensionality is high or dataset is dense, the cost of *BUC*-like algorithm increases dramatically.

## 6. CONCLUSIONS

In this paper, we have proposed and studied efficient methods for mining mutli-dimensional sequential patterns in large sequence databases. Multi-dimensional sequential patterns, which associate sequential patterns with multiple dimensional circumstance information, are interesting and useful in practice since people are often interested in detailed sequential patterns associated with different circumstances.

Taking *PrefixSpan* as our basic sequential pattern mining algorithm, and BUC as our basic multi-dimensional pattern mining algorithm, we have proposed and developed three algorithms, *UniSeq*, *Dim-Seq* and *Seq-Dim*, to incorporate additional dimensional information into the process of mining sequential patterns. *UniSeq* treats all dimension values as sequential items, finding all patterns using sequential pattern mining algorithm *PrefixSpan*, whereas the remaining two separate the mining of sequential items from other dimension values. The former, *Dim-Seq* , finds frequent dimension value combinations and then mines sequential patterns from the set of sequences that satisfy each of these combinations; whereas the latter, *Seq-Dim* , mines the sequential patterns for the whole dataset only once (using *PrefixSpan*), and then mines the corresponding frequent dimension patterns alongside each sequential pattern (using *BUC*). We investigate the strengths and limitations of each approach and show by experiments that *UniSeq* is the most effective when the total number of sequential items plus other dimension values is small; *Dim-Seq* is useful in datasets that are sparse with respect to dimension value combinations, but dense with respect to the sequential patterns present; and *Seq-Dim* is the better alternative in datasets that are dense with respect to both dimension value combinations and sequential items.

Multi-dimensional mining has been attracting attention in recent research into data mining [5]. We have been studying how to further improve the performance at mining multi-dimensional sequential patterns, how to mine efficiently max-sequential patterns, and closed-sequential patterns and how to incorporate user-specified constraints at mining such patterns. The applications of sequential pattern mining in Weblog analysis, telcommunication, biomedical research and DNA analysis are also interesting topics for furture research.

## 7. REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. 1995 Int. Conf. Data Engineering (ICDE'95)*, pages 3–14, Taipei, Taiwan, Mar. 1995.

[2] C. Bettini, X. Sean Wang, and S. Jajodia. Mining temporal relationships with multiple granularities in time sequences. *Data Engineering Bulletin*, 21:32–38, 1998.

[3] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In *Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99)*, pages 359–370, Philadelphia, PA, June 1999.

[4] M. Garofalakis, R. Rastogi, and K. Shim. Spirit: Sequential pattern mining with regular expression constraints. In *Proc. 1999 Int. Conf. Very Large Data Bases (VLDB'99)*, pages 223–234, Edinburgh, UK, Sept. 1999.

[5] G. Grahne, L. V. S. Lakshmanan, X. Wang, and M. H. Xie. On dual mining: From patterns to circumstances, and back. In *Proc. 2001 Int. Conf. Data Engineering (ICDE'01)*, pages 195–204, Heidelberg, Germany, April 2001.

[6] J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. In *Proc. 1999 Int. Conf. Data Engineering (ICDE'99)*, pages 106–115, Sydney, Australia, April 1999.

[7] J. Han, J. Pei, G. Dong, and K. Wang. Efficient computation of iceberg cubes with complex measures. In *Proc. 2001 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'01)*, Santa Barbara, CA, May 2001.

[8] H. Lu, J. Han, and L. Feng. Stock movement and n-dimensional inter-transaction association rules. In *Proc. 1998 SIGMOD Workshop Research Issues on Data Mining and Knowledge Discovery (DMKD'98)*, pages 12:1–12:7, Seattle, WA, June 1998.

[9] H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

[10] F. Masseglia, F. Cathala, and P. Poncelet. The psp approach for mining sequential patterns. In *Proc. 1998 European Symp. Principle of Data Mining and Knowledge Discovery (PKDD'98)*, pages 176–184, Nantes, France, Sept. 1998.

[11] B. Özden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Proc. 1998 Int. Conf. Data Engineering (ICDE'98)*, pages 412–421, Orlando, FL, Feb. 1998.

[12] J. Pei, J. Han, 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. 2001 Int. Conf. Data Engineering (ICDE'01)*, pages 215–224, Heidelberg, Germany, April 2001.

[13] S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. In *Proc. 1998 Int. Conf. Very Large Data Bases (VLDB'98)*, pages 368–379, New York, NY, Aug. 1998.

[14] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96)*, pages 1–12, Montreal, Canada, June 1996.

[15] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. 5th Int. Conf. Extending Database Technology (EDBT'96)*, pages 3–17, Avignon, France, Mar. 1996.

[16] J. Wang, G. Chirn, T. Marr, B. Shapiro, D. Shasha, and K. Zhang. Combinatiorial pattern discovery for scientific data: Some preliminary results. In *Proc. 1994 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'94)*, pages 115–125, Minneapolis, MN, May, 1994.

[17] M. J. Zaki. Efficient enumeration of frequent sequences. In *Proc. 7th Int. Conf. Information and Knowledge Management (CIKM'98)*, pages 68–75, Washington DC, Nov. 1998.