Improvements of IncSpan: Incremental Mining of Sequential Patterns in Large Database

Son N. Nguyen, Xingzhi Sun, and Maria E. Orlowska

School of Information Technology and Electrical Engineering,
The University of Queensland, QLD 4072, Australia
{nnson, sun, maria)@itee.uq.edu.au

Abstract. In reality, sequence databases are updated incrementally. The changes on the database may invalidate some existing sequential patterns and introduce new ones. Instead of recomputing the database each time, the incremental mining algorithms target efficiently maintaining the sequential patterns in the dynamically changing database.

Recently, a new incremental mining algorithm, called **IncSpan** was proposed at the International Conference on Knowledge Discovery and Data Mining (KDD'04). However, we find that in general, IncSpan fails to mine the *complete* set of sequential patterns from an updated database. In this paper, we clarify this weakness by proving the incorrectness of the basic properties in the IncSpan algorithm. Also, we rectify the observed shortcomings by giving our solution.

Keywords: Sequential patterns, Incremental mining, Algorithm.

1 Introduction

Discovering sequential patterns from databases is of great importance in many application domains (e.g., fault detection in a network, web access pattern analysis, and plan failure identification, etc.). Since the research problem of discovering sequential patterns was first introduced by Argawal et al. in [2], many mining algorithms [3, 4, 5, 6] have been proposed for efficiently finding frequent sequential patterns from the sequence databases.

In reality, a database is often dynamic. With the evolution of databases, some existing sequential patterns would be invalid and some new sequential patterns might be introduced. Thus, maintaining sequential patterns (over a significantly long period) becomes essential for sequential pattern mining.

Generally, the change on a sequential database can be categorized as 1) deleting records, 2) inserting new records, and 3) appending new items on the existing records. In this paper, discussions on the change of database only refer to the last two categories (i.e., we do not consider deleting records). Thus, the corresponding research problem of pattern maintenance can be described as follows. Given a sequence database D and the set FS of sequential patterns in D, when D evolves to D' (with the updated part db known, i.e., inserting and appending part), how to *efficiently* find the set FS' of sequential patterns in D' ?

T.B. Ho, D. Cheung, and H. Liu (Eds.): PAKDD 2005, LNAI 3518, pp. 442 – 451, 2005.

The naive approach to solve this problem is to apply a certain sequential mining algorithm A on D' to re-compute the sequential patterns. Obviously, it wastes computational resources because the previous mining result is not utilized in this subsequent problem. Another alternative, which will be discussed in this paper, is to incrementally update FS into FS'. The basic idea is to first investigate the new input data db while maximizing the use of a previous mining result FS. Various incremental update algorithms [7, 8, 9] have been designed to improve efficiency by reducing the number of scans on D'.

In general, an incremental update algorithm A' should satisfy the following two conditions.

1. Correctness: The mining result of A' should be identical to FS', where FS' is the set of frequent sequential patterns discovered by the traditional sequential mining algorithm A from the updated database D'.
2. Efficiency: Generally, for maintaining sequential patterns in a dynamically changing database, the time of applying A' is significantly less than that of re-computing D' by using A.

Recently, Cheng et al. [1] proposed a new incremental update algorithm, called IncSpan. IncSpan is developed on the basis of the sequential mining algorithm PrefixSpan. With the aims of improving performance, the redundant semi-frequent patterns (i.e., patterns that are "*almost frequent*" in D) are introduced and maintained as candidates of newly appearing sequential patterns in the updated database D'. In addition, some optimization techniques are applied in the pattern matching and projected database generation. The experimental results in [1] show that IncSpan significantly outperforms the non-incremental sequential mining algorithm [3] and the previously proposed incremental algorithm [7].

However, in this paper, we argue that in general, the algorithm IncSpan cannot find the complete set of frequent sequential patterns in the updated database D', i.e., it violates the correctness condition. Particularly, we give counter examples to prove that the foundations of IncSpan, and its three key properties presented in [1], are incorrect. The main purpose of this paper is to identify the weakness of IncSpan, and to propose a correct solution which can guarantee the completeness of the mining result.

The remainder of the paper is organised as follows. Section 2 first gives the formal definition of the incremental update problem and then describes the algorithm IncSpan. In Section 3, we prove that IncSpan fails to find the complete set of sequential patterns in the updated database. Based on IncSpan, our solution is proposed in Section 4. Finally, we conclude this paper in Section 5.

2 Summary of IncSpan Approach

For the completeness of this presentation and to establish our notation, we first formally define the problem of incremental sequential patterns mining.

2.1 Incremental Sequential Pattern Mining

Let $I = \{i_1, i_2, \ldots, i_k\}$ be a set of k distinct literals called items. A subset of I is called an itemset. A sequence $s = < t_1, t_2, \ldots, t_m>$ ($t_i \subseteq I$) is an ordered list. Without loss of

generality, we assume that the items in each itemset are sorted in certain order such as alphabetic order (a, b, c …). Formally, a sequence $\beta = <b_1, b_2, \ldots, b_n>$ is a subsequence of another sequence $\alpha = <a_1, a_2, \ldots, a_m>$, denoted $\beta \subseteq \alpha$, if and only if $\exists\ i_1, i_2, \ldots, i_n$ such that $1 \le i_1 < i_2 < \ldots < i_n \le m$ and $b_1 \subseteq a_{i1;} b_2 \subseteq a_{i2; \ldots;} b_n \subseteq a_{in.}$ In this case, we also call α is supersequence of β and α contains β.

A sequence database, $D = \{s_1, s_2, \ldots, s_n\}$, is a set of sequences. The support of a sequence α in D is the number of sequences in D which contain α, denoted as $sup_D(\alpha) = |\{s \mid s \in D \text{ and } \alpha \subseteq s\}|$. Given a support threshold, min_sup, a sequence is frequent if its support is no less than min_sup. The set of frequent sequential patterns, denoted as FS, includes all the frequent sequences in D.

The sequential pattern mining is to discover the complete set FS when the sequence database D and min_sup are given.

The incremental sequential pattern mining is formalized as follows. Given the original database D, db is the incremental part which is added to D. As the result, the new database is created, denoted as D' = D + db. There are two scenarios for this processing.

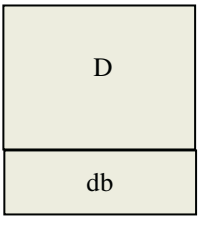

Fig. 2.1. Insert scenario

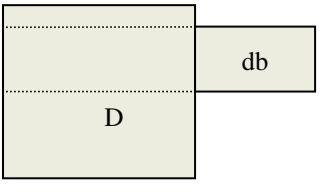

Fig. 2.2. Append scenario

The Insert scenario (Figure 2.1) means that the new sequences are inserted into the original database. These old sequences are still unchanged, but the total number of the sequences in D' is increased. In the Append scenario (Figure 2.2), some old sequences are appended with new sequences. And the total number of the sequences in D' is unchanged. If we consider Insert scenario is a special case of Append scenario (i.e., db is appended with the empty sequences), we can combine two scenarios and formulate the problem as follows.

Given a sequence $s = < t_1, t_2, \ldots, t_m> \in D$ and another sequence $s_a = < t'_1, t'_2, \ldots, t'_n> \in$ db, if s concatenates with s_a in D', the new sequence s' is called an appended sequence, denoted as $s' = s + s_a$. Obviously, s' is always not empty. If we assume that s is empty, then we can treat the Insert scenario as the Append scenario. We denote LDB = {s' | s' ∈ D' and s' is appended with items/itemsets}. We denote ODB = {s | s ∈ D and s is appended with items/itemsets in D'}, denote NDB = {s | s ∈ D and s is not appended with items/itemsets in D'}. The set of frequent sequences in D' is denoted as FS'. As a result, we have the following formula.

D' = D + db = (ODB + NDB) + db = (ODB + db) + NDB = LDB + NDB.

Example 2.1: A sample sequence database D and an appended part db are given to explain the notations defined above

Table 2.1. A sample sequence database D and the appended part

Seq ID	Original Part	Appended Part
0	(a)(h)	***(c)***
1	(eg)	***(a)(bce)***
2	(a)(b)(d)	***(ck)(l)***
3	(b)(df)(a)(b)	***∅***
4	(a)(d)	***∅***
5	(be)(d)	***∅***

Given min_sup = 3, we have:
ODB = {SeqID: 0-2 in D}; NDB = {SeqID:3-5 in D}; LDB = {SeqID: 0-2 in D'};
FS={<(a)>:4, <(b)>:3, <(d)>:4, <(b)(d)>:3};

FS'={<(a)>:5, <(b)>:4, <(d)>:4, <(b)(d)>:3, <(c)>:3, <(a)(b)>:3, <(a)(c)>:3};

Problem statement: Given D, D', min_sup, and FS, the incremental sequential pattern mining is to mine the set FS' in D' based on FS , rather than to recompute D' from scratch.

2.2 IncSpan Approach

In this part, we introduce IncSpan [1], the recently proposed approach for the incremental sequential pattern mining. In [1], the authors present the idea of buffering semi-frequent patterns as the following description, study its properties, and design solutions of how to mine and maintain FS incrementally based on PrefixSpan approach.

Buffering Semi-frequent Pattern: Given a factor $\mu \leq 1$, a sequence is semi-frequent if its support is less than min_sup but no less than μ * min_sup; a sequence is infrequent if its support is less than μ * min_sup. The set of semi-frequent sequential pattern includes all the semi-frequent sequences in D and D', denoted as SFS and SFS' respectively. For example, given μ = 0.6, according to Example 2.1, we have SFS={<(e)>:2, <(a)(b)>:2, <(a)(d)>:2}and SFS'={<(e)>:2, <(a)(d)>:2, <(be)>:2}.

PrefixSpan Approach: PrefixSpan approach [3] is one of the most efficient methods for mining sequential patterns. It uses prefix sequence, suffix sequence and p-projected database concepts to discover the complete set of sequential patterns when the sequence database D and min_sup are given, more detail can refer to [3]. The prototype of PrefixSpan algorithm which is used in the following sections is **PrefixSpan(p, D|p, μ * min_sup, FS, SFS),** where p is a sequence, D|p is p-projected database of D. This routine is called recursively to mine the complete FS, SFS in D|p.

It is assumed in [1] that the sequences in SFS are "almost frequent" in D, most of the frequent subsequences in the appended database (D') will either come from SFS

or they are already frequent in original database D. According to [1], the SFS' and FS" in D' are derived from the following cases:

1. A pattern p which is frequent in D, is still frequent in D'
2. A pattern p which is semi-frequent in D, becomes frequent in D'
3. A pattern p which is semi-frequent in D, still semi-frequent in D'
4. Appended data db brings new frequent / semi-frequent items
5. A pattern p which is infrequent in D, becomes frequent in D'
6. A pattern p which is infrequent in D, becomes semi-frequent in D'

Case (1) - (3). There exsists information in FS and SFS, so one can update the support number and project D' to find all frequent / semi-frequent sequences which are generated from FS and SFS.

Case (4). Property 2.1: An item which does not appear in D and is brought by db has no information in FS and SFS. **Solution 2.1:** Scan LDB for single items. Then use the new frequent item as prefix to construct projected database and discover frequent sequences recursively by PrefixSpan approach.

Case (5). Property 2.2: An infrequent sequence p' in D becomes frequent in D', all of its subsequences must also be frequent in D'. Then at least one of its prefix subsequences, p, is in FS. **Solution 2.2:** Start from its prefix p in FS and construct p-projected database on D', use PrefixSpan approach, IncSpan will discover p'.

IncSpan provides the pruning technique based on the following theorem.

Theorem 2.1. *For a frequent pattern p, if its support in LDB* $sup_{LDB}(p) < (1 - \mu) * min\_sup$, *then there is no sequence p' having p as prefix changing from infrequent in D to frequent in D'.*

This theorem provides an effective bound to decide whether it is necessary to project the whole database D', which can reduce the number of projections.

Case (6). Property 2.3: An infrequent sequence p' in D becomes semi-frequent in D', all of its subsequences must also be semi-frequent in D'. Then at least one of its prefix subsequences, p, is in FS or SFS. **Solution 2.3:** Start from its frequent prefix p in FS or SFS, and construct p-projected database on D', use PrefixSpan approach, IncSpan will discover p'.

IncSpan Algorithm Outline: Given an original database D, an appended database D', a threshold min_sup, a buffer ratio μ, a set of frequent sequences FS and a set of semi-frequent sequences SFS, IncSpan wants to discover only the set of frequent sequences FS' in D' [1]. The basic idea of algorithm are described as follows.

Step 1: Scan LDB for single items, as show in case (4).

Step 2: Check every pattern in FS and SFS in LDB to adjust the support.

Step 2.1: If a pattern becomes frequent, add it to FS'. Then check whether it meets the projection condition according to Theorem 1. If so, use it as prefix to project database D', as show in case (5).

Step 2.2: If a pattern is semi-frequent, add it to SFS'.

Algorithm Outline: IncSpan(D', min_sup, μ, FS, SFS)
Input: An appended database D', min_sup, FS and SFS in D
Output: FS' in D'
Method:

```
1:   FS' = ∅ ; SFS' = ∅ ;
2:   Scan the LDB for new single items
3:         Add new frequent items into FS'
4:         Add new semi-frequent items into SFS'
5:   For each new item i in FS' do
6:         PrefixSpan(i, D'|i, μ * min_sup, FS', SFS')
7:   For every pattern p in FS or SFS do
8:         Check Δsup(p) = sup_db(p)
9:         If sup_D'(p) = sup_D(p) + Δsup(p) ≥ min_sup
10:              Insert(FS', p)
11:              If supLDB(p) ≥ (1 - μ) * min_sup
12:                   PrefixSpan(p, D'|p, μ * min_sup, FS', SFS')
13:        Else
14:              Insert(SFS', p)
15:  Return;
```

3 Critical Observations

After the IncSpan approach is reviewed in Section 2, in this section, we show that in general, IncSpan provides incomplete results. Particularly, we prove that the solutions in IncSpan for Case 4-6 are incorrect by giving counter examples.

Claim 3.1 (Incorrectness of Solution 2.1 for **Case (4)):** Scanning LDB cannot find the complete set of new single frequent / semi-frequent items in D'.

Proof: Generally, scanning LDB can only discover the new single frequent / semi-frequent items in terms of LDB. Since the support is counted as number, for the single items that are infrequent ones in LDB and D but become frequent / semi-frequent in D', Solution 2.1 fails to discover them. The following example illustrates the incompleteness of frequent single items. The example for semi-frequent single items can be created by following the same idea.

Counter example 3.1: This example is generated from Example 2.1 with small change in appended part: item (f) is appended in SeqID 0-1. Remember that min_sup is 3 and μ is 0.6. If the IncSpan algorithm scans only LDB, the new frequent item (f) in D' cannot be discovered. As a result, IncSpan loses all new frequent sequences which have (f) as a prefix.

Claim 3.2 (Incorrectness of Property 2.2 for Case (5)): In IncSpan, if an infrequent sequence p' in D becomes frequent in D', it is possible that none of its prefix subsequence p is in FS.

Proof: A counter example is illustrated as follows.

Counter example 3.2: This example generates from Example 2.1 with a small change in original part (SeqID 3-5 are deleted). Given min_sup = 3; μ = 0.6

In this example, <(a)(c)> is infrequent in D but becomes frequent in D'. However, its prefix subsequences, <(a)>, is not in FS, in this case, FS is a empty set.

Claim 3.3 (Incorrectness of Property 2.3 for Case (6)): In IncSpan, if an infrequent sequence p' in D becomes semi-frequent in D', it is possible that none of its prefix subsequence p is in FS or SFS.

Proof: A counter example is illustrated as follows.

Counter example 3.3: This example generates from Example 2.1 with a small change in original part, and appended part. Given min_sup = 3; μ = 0.6

In this example, <(be)> is infrequent in D but becomes semi-frequent in D'. However, its prefix subsequences, <(b)>, is not in SFS. In this case, SFS = {<(a)>:2, <(e)>:2}.

Table 3.1. A sample sequence database D and the new appended part

Seq ID	Original Part	Appended Part
0	(a)(h)	***(c)(f)***
1	(eg)	***(a)(bce)(f)***
2	(a)(b)(d)	***(ck)(l)***
3	(b)(df)(a)(b)	***∅***
4	(a)(d)	***∅***
5	(be)(d)	***∅***

Table 3.2. A deleted sequence database D and the appended part

Seq ID	Original Part	Appended Part
0	(a)(h)	***(c)***
1	(eg)	***(a)(bce)***
2	(a)(b)(d)	***(ck)(l)***

Table 3.3. A deleted sequence database D and the new appended part

Seq ID	Original Part	Appended Part
0	(a)(h)	***(c)***
1	(eg)	***(a)(bce)***
2	(a)(be)(d)	***(ck)(l)***

Claim 3.4 (Extension of Theorem 2.1): In order to apply the pruning technique based on Theorem 2.1 for any pattern p in FS or SFS. Theorem 2.1 can be extended as follows. The difference between the following theorem and Theorem 2.1 is that we can apply for not only frequent pattern p in D, but also any other pattern p in D.

Theorem 3.1: *For any pattern p in D, if its support in LDB* $sup_{LDB}(p) < (1 - \mu) * min\_sup$, *then there is no sequence p' having p as prefix changing from infrequent in D to frequent in D'.*

Proof: p' was infrequent in D, so $sup_D(p') < \mu * min\_sup$ (i)

If $sup_{LDB}(p) < (1 - \mu) * min\_sup$ then

$sup_{LDB}(p') \leq sup_{LDB}(p) < (1 - \mu) * min\_sup$ because $p \subset p'$.

Since $sup_{LDB}(p') = sup_{ODB}(p') + sup_{db}(p')$ because LDB = ODB + db, then

$sup_{db}(p') \leq sup_{LDB}(p') < (1 - \mu) * min\_sup$ (ii)

Combining (i) and (ii), we have $sup_{D'}(p') = sup_D(p') + sup_{db}(p') < min\_sup$. So p' cannot be frequent in D'.

4 Proposed the Complete Solution

With all above observations in mind, we present the following algorithm for improvement of IncSpan, denoted as IncSpan+.

Given an original database D, an appended database D', a minimum support min_sup, a buffer ratio μ, a set of frequent sequences FS and a set of semi-frequent sequences SFS, we want to mine the set of frequent sequences FS', and the set of semi-frequent sequences SFS' in D'.

Algorithm Outline: IncSpan+(D', min_sup, μ, FS, SFS)
Input: An appended database D', min_sup, FS and SFS in D
Output: FS', SFS' in D'
Method:

```
1:  FS' = ∅ ; SFS' = ∅ ;
2:  Determine LDB; Total number of sequences in D', adjust the min_sup if it is
    changed due to the increasing of total number of sequences in D'.
3:  Scan the whole D' for new single items
4:          Add new frequent items into FS'
5:          Add new semi-frequent items into SFS'
6:  For each new item i in FS' do
7:          PrefixSpan(i, D'|i, μ * min_sup, FS', SFS')
8:  For each new item i in SFS' do
9:          PrefixSpan(i, D'|i, μ * min_sup, FS', SFS')
10: For every pattern p in FS or SFS do
11:         Check Δsup(p) = sup_db(p)
12:         If sup_D'(p) = sup_D(p) + Δsup(p) ≥ min_sup
13:                 Insert(FS', p)
14:                 If supLDB(p) ≥ (1 - μ) * min_sup
15:                         PrefixSpan(p, D'|p, μ * min_sup, FS', SFS')
16:         ElseIf sup_D'(p) ≥ μ * min_sup
17:                 Insert(SFS', p)
18:                 PrefixSpan(p, D'|p, μ * min_sup, FS', SFS')
19: Return;
```

This algorithm follows the same spirit as IncSpan, using PrefixSpan approach to maintain both FS' and SFS'. However, new proposed algorithm ensures the correctness of mining result in the updated database, as is proven in Claim 4.1.

Claim 4.1 (Correctness of IncSpan+): The IncSpan+ outputs the complete set of frequent pattern FS' and the complete set of semi-frequent pattern SFS' in the updated database D'.

Proof: The complete FS' and SFS' sets come exactly from two sources:

Case (4.1): From new frequent / semi-frequent single items and all of frequent / semi-frequent supersequences which have these new single items as the prefix;

Case (4.2): From FS, SFS, and sequences that have the prefix in FS or SFS.

As can be seen from the outline of IncSpan+, line from 3-9 will discover all frequent sequences and semi-frequent sequences in D' corresponding to case (4.1). That means all frequent sequences / semi-frequent sequences in D', which have the first new frequent / semi-frequent item as the prefix, will be discovered. These new frequent / semi-frequent items are not included in FS and SFS.

Line 10-18 in IncSpan+ will discover all frequent / semi-frequent sequences which have their prefix in FS or SFS, and all of them correspond to case (4.2).

Claim 4.1 proves the correctness of our proposed IncSpan+. Compared with the original approach, IncSpan+ has the following improvements:

1. IncSpan+ can find the complete FS', which guarantees the correctness of the mining result.
2. IncSpan+ can find the complete SFS', which is helpful in incrementally maintaining the frequent patterns for further database updates.

5 Conclusion

This paper clarified the weakness of the recent work [1] in the context of the incremental mining sequential patterns. We proved that IncSpan, the incremental mining approach in [1], cannot find the complete set of sequential patterns in the updated database. The solution, IncSpan+ was proposed to rectify the observed shortcoming. IncSpan+ not only guarantees the correctness of the incremental mining result, but maintains the complete set of semi-frequent sequences for future updates.

References

1. Cheng H., Yan X., Han J.: IncSpan: Incremental mining of sequential patterns in large database. Proc. ACM KDD Conf. on Knowledge Discovery in Data, Washington (KDD'04), 2004
2. Agrawal R., Srikant R.: Mining sequential patterns. Proc. 11th IEEE Int. Conf. on Data Engineering (ICDE'95), 1995
3. Pei J., Han J., Mortazavi-Asl B., Wang J., Pinto H., Chen Q., Dayal U., Hsu M.: Mining sequential patterns by Pattern-Growth: The PrefixSpan approach. IEEE Transactions on Knowledge and Data Engineering, Vol.16, No. 10, 2004
4. Srikant R., Agrawal R.: Mining sequential patterns: Generalizations and performance improvements. Proc. 5th IEEE Int. Conf. on Extending Database Technology (IDBT'96)

5. Zaki M.: SPADE: An efficient algorithm for mining frequent sequences, Machine Learning, 40: (31-60), 2001
6. Ayres J., Gehrke J.E., Yiu T., Flannick J.: Sequential pattern mining using bitmaps. Proc. ACM KDD Conf. on Knowledge Discovery in Data (KDD'02), 2002
7. Parthasarathy S., Zaki M., Ogihara M., and Dwarkadas S.: Incremental and interactive sequence mining. Proc. 8th Int. Conf. on Information and Knowledge Management (CIKM'99), 1999
8. Masseglia F., Poncelet P., Teisseire M.: Incremental mining of sequential patterns in large database. Data & Knowledge Engineering, 46: (97-121), 2003
9. Zhang M., Kao B., Cheung D., Yip C.: Efficient algorithms for incremental update of frequent sequences. Proc. of Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'02), 2002