ApproxMAP: Approximate Mining of Consensus Sequential Patterns

Hye-Chung (Monica) Kum[†] **Jian Pei**[‡] **Wei Wang**[†] **Dean Duncan**[†]

[†] University of North Carolina at Chapel Hill, {kum, weiwang}@cs.unc.edu, dfduncan@email.unc.edu

[‡] State University of New York at Buffalo, jianpei@cse.buffalo.edu

Abstract

Conventional sequential pattern mining methods may meet inherent difficulties in mining databases with long sequences and noise. They may generate a huge number of short and trivial patterns but fail to find interesting patterns approximately shared by many sequences. In this paper, we propose the theme of *approximate sequential pattern mining* roughly defined as *identifying patterns approximately shared by many sequences*. We present an efficient and effective algorithm, ApproxMAP, to mine consensus patterns from large sequence databases in two steps. First, sequences are clustered by similarity. Then, consensus patterns are mined from each cluster through multiple alignment. We use a real case study to illustrate the effectiveness of ApproxMAP.

1 Introduction

A sequential pattern is a subsequence that appears frequently in a sequence database. Since it has been proposed in [1], mining sequential patterns in large databases has become an important data mining task with broad applications, such as business analysis, web mining, security, and bio-sequences analysis.

Although sequential pattern mining has been extensively studied and many methods have been proposed (e.g., GSP [12], SPADE [13], PrefixSpan [10], FreeSpan [6], and SPAM [2]), there are two inherent obstacles within the conventional framework.

First, *most methods mine sequential patterns with exact matching*. A pattern is supported by sequences in the database only if the pattern *exactly appears* in the sequences. However, the exact matching approach often may not find general long patterns in the database. For example, many customers may share similar buying habits, but few follow the exact same buying pattern. Thus, to find non-trivial interesting long patterns, we must consider mining *approximate* sequential patterns.

Second, *most methods mine the complete set of sequential patterns*. When long patterns exist, mining the complete set of patterns is ineffective and inefficient, since every sub-pattern of a long pattern is also a pattern. In many situations, a user may just want the *long patterns* that cover many short ones. Recently, mining compact expressions for frequent patterns, such as max-patterns [3] and frequent closed patterns [9], has been proposed and studied in the context of frequent itemset mining. However, mining max-sequential patterns or closed sequential patterns is far from trivial. Furthermore, in a noisy sequence database, the number of max- or closed sequential patterns still can be huge, and many of them are trivial for users.

In this paper, we propose an effective and efficient framework for mining approximate sequential patterns in databases of long sequences, and make the following contributions. We propose the theme of *approximate sequential pattern mining*. The general idea is that, instead of finding exact patterns, we identify patterns approximately shared by many sequences. Instead of mining a huge set of patterns, we propose to mine *consensus patterns* from databases of long sequences. Intuitively, a consensus pattern is shared by many sequences and covers many short patterns. Consensus patterns are more expressive and are more useful in many applications. We develop an efficient algorithm, ApproxMAP (for APPROXimate Multiple Alignment Pattern mining), to mine consensus sequential patterns from large databases. ApproxMAP finds the underlying consensus patterns from multiple alignment directly. It is effective and efficient for mining long sequences and is robust to noise. We use a real case study to illustrate the effectiveness of ApproxMAP.

The remainder of the paper is organized as follows. Section 2 defines the problem. Section 3 and 4 demonstrate the ApproxMAP method in detail. A successful case study is reported in Section 5. Section 6 concludes the paper.

2 Problem Statement

Let $I = \{i_1, \ldots, i_l\}$ be a set of *items*. An *itemset* $X = \{i_{j_1}, \ldots, i_{j_k}\}$ is a subset of I. Conventionally, itemset $X = \{i_{j_1}, \ldots, i_{j_k}\}$ is also written as $(x_{j_1} \cdots x_{jk})$. A *sequence* $S = \langle X_1 \ldots X_n \rangle$ is an ordered list of itemsets, where $X_1, \ldots, X_n$ are all itemsets. A *sequence database* SDB is a multi-set of sequences.

A sequence $S_1 = \langle X_1 \cdots X_n \rangle$ is called a *subsequence* of sequence $S_2 = \langle Y_1 \cdots Y_m \rangle$, and S_2 a *super-sequence* of S_1, if $n \leq m$ and there exist integers $1 \leq i_1 < \cdots < i_n \leq m$ such that $X_j \subseteq Y_{i_j}$ $(1 \leq j \leq n)$. Given a sequence

database SDB, the *support* of a sequence P, denoted as $sup(P)$, is the number of sequences in SDB that are super-sequences of P. Conventionally, a sequence P is called a *sequential pattern* if $sup(P) \geq min\_sup$, where min_sup is a user-specified *minimum support threshold*.

In many applications, people prefer long sequential patterns *approximately* shared by many sequences. Motivated by this observation, we introduce the notion of *mining approximate sequential patterns*. Let $dist$ be a normalized distance measure of two sequences with domain $[0, 1]$. For sequences S, S_1 and S_2, if $dist(S, S_1) < dist(S, S_2)$, then S_1 is said be *more similar* to S than S_2 is.

Naïvely, we can extend the conventional sequential pattern mining framework to an approximate sequential pattern mining framework as follows. Given a *minimum distance threshold* min_dist, the *approximate support* of a sequence P in a sequence database SDB is defined as $\widetilde{sup}(P) = \|\{S|(S \in SDB) \wedge (dist(S, P) \leq min\_dist)\}\|$. (Alternatively, the approximate support can be defined as $\widetilde{sup}(P) = \sum_{S \in SDB} dist(S, P)$. All the following discussion retains.) Given a minimum support threshold, min_sup, all sequential patterns whose approximate support pass the threshold can be mined. However, the naïve method could suffer from the following problems.

First, *the mining may find many short and probably trivial patterns*, since short patterns tend to be easier to get similarity counts from the sequences than long patterns. Second, *the complete set of approximate sequential patterns may be larger than that of exact sequential patterns and thus difficult to understand.* By approximation, a user may want to get and understand the general trend and ignore the noise. However, a naïve output of the complete set of approximate patterns in the above framework may generate many (trivial) patterns and thus ruin the mining.

We need to explore a more effective solution. We propose ApproxMAP, a *cluster and multiple alignment-based* approach, which works in two steps. Firstly, sequences in a database are clustered based on similarity. Sequences in the same cluster may approximately follow some similar patterns. Then, the longest approximate sequential pattern for each cluster is generated. It is called the *consensus pattern*. To extract consensus patterns, a weighted sequence is derived for each cluster using multiple alignment to compress the sequential pattern information in the cluster. And then the longest consensus pattern best representing the cluster is generated from the weighted sequence.

3 Clustering Sequences

In general, the *hierarchical edit distance* is commonly used as a distance measure for sequences. It is defined as the minimum cost of editing operations (i.e., insertions, deletions, and replacements) required to change one sequence to the other. An insertion operation on S_1 to change it towards S_2 is equivalent to a deletion operation on S_2 towards S_1. Thus, an insertion operation and a deletion operation have the same cost. We use $INDEL()$ to denote an insertion or deletion operation, and $REPL()$ to denote a replacement operation. Often, the following inequality is assumed.

$$REPL(X, Y) \leq INDEL(X) + INDEL(Y)$$

Given two sequences $S_1 = \langle X_1 \cdots X_n \rangle$ and $S_2 = \langle Y_1 \cdots Y_m \rangle$, the hierarchical edit distance between S_1 and S_2 can be computed by dynamic programming using the following recurrence relation.

$$\begin{aligned} &D(0,0)=0 \\ &D(i,0)=D(i-1,0)+INDEL(X_i) \text{ for } (1 \leq i \leq n) \\ &D(0,j)=D(0,j-1)+INDEL(Y_j) \text{ for } (1 \leq j \leq m) \\ &D(i,j)=\min \begin{cases} D(i-1,j)+INDEL(X_i) \\ D(i,j-1)+INDEL(Y_j) \\ D(i-1,j-1)+REPL(X_i,Y_j) \end{cases} \\ &\qquad \text{for } (1 \leq i \leq n) \text{ and } (1 \leq j \leq m) \end{aligned} \tag{3.1}$$

To make the edit distances comparable between sequences with various lengths, we normalize the results by dividing the hierarchical edit distance by the length of the longer sequence in the pair, and call it the *normalized edit distance*. That is,

$$dist(S_1, S_2) = \frac{D(n, m)}{\max\{\|S_1\|, \|S_2\|\}} \tag{3.2}$$

To extend the hierarchical edit distance to sequences of sets, we need to define the cost of edit operations (i.e., INDEL() and REPL() in Equation 3.1) properly. Here, we adopt the *normalized set difference* as the cost of replacement of sets.

$$\begin{aligned} REPL(X, Y) &= \frac{\|(X-Y)\cup(Y-X)\|}{\|X\|+\|Y\|} \\ &= \frac{\|X\|+\|Y\|-2\|X\cap Y\|}{\|X\|+\|Y\|} \end{aligned} \tag{3.3}$$

This measure is a metric [4] and has a nice property that $0 \leq REPL() \leq 1$. $REPL()$ is mathematically equivalent to the Sørensen coefficient, an index similar to the Jaccard coefficient except that it gives more weight to the common elements [8]. Thus, it is more appropriate if the commonalities are more important than the differences.

Using the hierarchical edit distance (Equation 3.2), we can apply a density-based clustering algorithm to cluster sequences. Intuitively, a sequence is "*dense*" if there are many sequences similar to it in the database. In particular, for each sequence S_i in a database $\mathcal{S}$, let $d_1, \ldots, d_k$ be the k smallest non-zero values of $dist(S_i, S_j)$, where $S_j \neq S_i$, is a sequence in $\mathcal{S}$. Then,

$$\begin{aligned} &Density(S_i) = \frac{n}{\|\mathcal{S}\| d} \\ \text{where} \quad &d = \max\{d_1, \ldots, d_k\} \\ \text{and} \quad &n = \|\{S_j \in \mathcal{S} | dist(S_i, S_j) \leq d\}\|. \end{aligned} \tag{3.4}$$

In Equation 3.4, n is the number of sequences in the k-nearest neighbor space (including all ties). Here k is a user-specified parameter.

We adopt an algorithm from [11] as follows.

ALGORITHM 3.1. **Uniform kernel k-NN clustering**

Input: a set of sequences $\mathcal{S} = \{S_i\}$, # of neighbor sequences k;

Output: a set of clusters $\{C_j\}$, where each cluster is a set of sequences;

Method:

Step 1: Initialize every sequence as a cluster. For each sequence S_i in cluster C_{S_i}, set $Density(C_{S_i}) = Density(S_i)$.

Step 2: Merge nearest neighbors based on the density of sequences. For each sequence S_i, let $S_{i_1}, \ldots, S_{i_n}$ be the nearest neighbor of S_i, where n is defined in Equation 3.4. For each $S_j \in \{S_{i_1}, \ldots, S_{i_n}\}$, merge cluster C_{S_i} containing S_i with a cluster C_{S_j} containing S_j, if $Density(S_i) < Density(S_j)$ and there exists no S'_j having $dist(S_i, S'_j) < dist(S_i, S_j)$ and $Density(S_i) < Density(S'_j)$. Set the density of the new cluster to $\max\{Density(C_{S_i}), Density(C_{S_j})\}$.

Step 3: Merge based on the density of clusters. For all sequences S_i such that S_i has no nearest neighbor with density greater than that of S_i, but has some nearest neighbor, S_j, with density equal to that of S_i, merge the two clusters C_{S_j} and C_{S_i} containing each sequence if $Density(C_{S_j}) > Density(C_{S_i})$. This step is to merge "*plateau neighbor regions*". ■

It is easy to show that the above algorithm has complexity $O(kN_{seq})$. The key parameter for the clustering process in Algorithm 3.1 is k, the number of nearest neighbors that the algorithm will search. A larger k value tends to merge more sequences, and results in a smaller number of large clusters, while a smaller k value tends to break up clusters. The benefit of using a small k value is that the algorithm can detect less frequent patterns. The tradeoff is that it may break up clusters representing strong patterns to generate multiple similar patterns. As shown in our performance study, in many applications, a value of k in the range from 3 to 10 works well [7].

4 Multiple Alignment and Pattern Generation

Once sequences are clustered, sequences within a cluster are similar to each other. Now, the problem becomes *how to summarize the general pattern in each cluster and discover the trend*. In this section, we develop a method using multiple alignment. First, we discuss how to align sequences in a cluster, then we explore how to summarize sequences and generate patterns.

4.1 Multiple Alignment of Sequences

In general, for a cluster C with n sequences $S_1, \ldots, S_n$, finding the *optimal global alignment* that minimizes $\sum_{j=1}^{n} \sum_{i=1}^{n} dist(S_i, S_j)$ is an NP-hard problem [5], and thus is impractical for mining large sequence databases with many sequences.

In a cluster, some sequences may be similar to many other sequences in the cluster. These sequences are most likely to be closer to the underlying patterns than the other sequences. It is more likely to get an alignment close to the optimal one, if we start the alignment with such "*seed*" sequences. Intuitively, the *density* defined in Equation 3.4 measures the similarity between a sequence and its nearest neighbors. Thus, a sequence with a high density means that it has some neighbors very similar to it, and it is a good candidate for a "*seed*" sequence in the alignment.

As the first step in the clustering (see Algorithm 3.1), the density for each sequence is calculated. We only need to sort all sequences within a cluster in density descending order.

To store the alignment results effectively, we propose a notion of weighted sequence as follows. A weighted sequence WS=$\langle X_1 : v_1, \ldots, X_l : v_l\rangle : n$ carries the following information:

1. the current alignment has n sequences, and n is called the *global weight* of the weighted sequence;
2. in the current alignment, v_i sequences have a non-empty itemset X_i aligned in the i^{th} itemset, where $(1 \le i \le l)$;
3. an itemset in the alignment is in the form of $X_i = (x_{j_1} : w_{j_1}, \ldots, x_{j_m} : w_{j_m})$, which means, in the current alignment, there are w_{j_k} sequences that have item x_{j_k} in the i^{th} position of the alignment, where $(1 \le i \le l)$ and $(1 \le k \le m)$.

We illustrate how to use weighted sequences to do multiple alignment in the following example.

Example 1. (Multiple alignment) Suppose that, in a cluster C, there are 5 sequences as shown in Table 1. The density descending order of these sequences is S_3-S_2-S_4-S_5-S_1. The sequences are aligned as follows.

First, sequences S_3 and S_2 are aligned as shown in Figure 1. Here, a *weighted sequence* WS_1 is used to summarize and compress the information about the alignment. Since the first itemsets of S_3 and S_2, (a) and (ae), are aligned in the same position, the first itemset in the weighted sequence WS_1 is $(a : 2, e : 1) : 2$. It means that two sequences are aligned in this position, and a and e appear twice and once, respectively. The second itemset in WS_1, $(h : 1) : 1$, means there is only one sequence with an itemset aligned in this position, and item h appears once.

S_3	$\langle(a)$		(b)	$(de)\rangle$	
S_2	$\langle(ae)$	(h)	(b)	$(d)\rangle$	
WS_1	$\langle(a:2, e:1):2$	$(h:1):1$	$(b:2):2$	$(d:2, e:1):2\rangle$	: 2

Figure 1: S_3 and S_2 are aligned resulting in WS_1.

After the first step, we need to iteratively align other sequences with the current weighted sequence. The weighted sequence does not explicitly keep information about various itemsets in the sequences. Instead, this information is summarized into the item weights in the weighted sequence which need to be taken into account when aligning a sequence to a weighted sequence. Thus, we adopt a *weighted replace cost* as follows.

Let $X = (x_1 : w_1, \ldots, x_m : w_m) : v$ be an itemset in a weighted sequence, while $Y = (y_1 \cdots y_l)$ is an itemset in a sequence in the database. Let n be the global weight of the weighted sequence. The replace cost is defined as

$$REPL(X, Y) = \frac{e_R \cdot v + n - v}{n} \tag{4.5}$$

$$\text{where} \quad e_R = \frac{\sum_{i=1}^{m} w_i + \|Y\| v - 2\sum_{x_i \in Y} w_i}{\sum_{i=1}^{m} w_i + \|Y\| v}$$

Seq-id	Sequence	Alignment				
S_1	$\langle(ag)(f)(bc)(ae)(h)\rangle$	$\langle(ag)$	(f)	(bc)	(ae)	$(h)\rangle$
S_2	$\langle(ae)(h)(b)(d)\rangle$	$\langle(ae)$	(h)	(b)	(d)	$\rangle$
S_3	$\langle(a)(b)(de)\rangle$	$\langle(a)$		(b)	(de)	$\rangle$
S_4	$\langle(a)(bcg)(d)\rangle$	$\langle(a)$		(bcg)	(d)	$\rangle$
S_5	$\langle(bci)(de)\rangle$	$\langle$		(bci)	(de)	$\rangle$
Weighted sequence		$\langle(a:4,e:1,g:1):4$	$(f:1,h:1):2$	$(b:5,c:3,g:1,i:1):5$	$(a:1,d:4,e:3):5$	$(h:1):1\rangle:5$

Table 1: Sequences in a cluster and the complete alignment.

Accordingly, we have $INDEL(X) = REPL(X, \emptyset) = 1$ and $INDEL(Y) = REPL(Y, \emptyset) = 1$.

In the next step, the weighted sequence WS_1 and the third sequence S_4 are aligned as shown in Figure 2. Similarly, we can align the remaining sequences. The results are shown in Figure 3.

The alignment result for all sequences are summarized in the weighted sequence WS_4 shown in Figure 3. After the alignment, we only need to store WS_4. All the sequences in the cluster are not needed any more in the remainder of the mining. ■

Aligning the sequences in different order may result in slightly different weighted sequences. As verified by our extensive empirical evaluation, the alignment order has minor effect on the underlying patterns[7].

4.2 Generation of Consensus Patterns

As shown in Section 4.1, a weighted sequence records the statistics of the alignment of the sequences in a cluster. Intuitively, a pattern can be generated by picking up parts of a weighted sequence shared by most sequences in the cluster.

For a weighted sequence $WS = \langle(x_{11} : w_{11}, \ldots, x_{1m_1} : w_{1m_1}) : v_1, \ldots, (x_{l1} : w_{l1}, \ldots, x_{lm_l} : w_{lm_l}) : v_l\rangle : n$, the *strength* of item $x_{ij} : w_{ij}$ in the i^{th} itemset is defined as $\frac{w_{ij}}{n} \cdot 100\%$. Clearly, an item with a larger strength value indicates that the item is shared by more sequences in the cluster.

Motivated by the above observation, a user can specify a *strength threshold* ($0 \leq min\_strength \leq 1$). A *consensus pattern* P can be extracted from a weighted sequence by removing items in the sequence whose strength values are lower than the threshold.

Example 2. (Consensus pattern generation) Suppose a user specifies a strength threshold $min\_strength = 30\%$. The consensus pattern extracted from weighted sequence WS_4 is $\langle(a)(bc)(de)\rangle$.

Interestingly, if we compare the sequences in the sequence database (Table 1) and the consensus pattern mined from the database, the pattern is shared by the sequences, but it is not exactly contained in any one of them. In particular, every sequence except S_2 approximately contains the pattern by one insertion. These evidences strongly indicate that the consensus pattern is the general template behind the data. ■

5 Case Study:Mining The Welfare Services DB

An extensive performance evaluation of ApproxMAP verifies that ApproxMAP is both effective and efficient[7]. Limited by space, we omit the details here. Instead, we report the result on a real data set of welfare services accumulated over a few years in North Carolina State. The services have been recorded monthly for children who had a substantiated report of abuse and neglect, and were placed in foster care. There were 992 such sequences. In summary we found 15 interpretable and useful patterns.

As an example, in total 419 sequences were grouped together into one cluster which had the following consensus pattern.

$$\langle(RPT)(INV, FC)\overbrace{(FC)\cdots(FC)}^{11}\rangle$$

In the pattern, RPT stands for a report, INV stands for an investigation, and FC stands for a foster care service. The pattern indicates that many children who are in the foster care system after getting a substantiated report of abuse and neglect have very similar service patterns. Within one month of the report, there is an investigation and the child is put into foster care. Once children are in the foster care system, they stay there for a long time. This is consistent with the policy that all reports of abuse and neglect must be investigated within 30 days. It is also consistent with our analysis on the length of stay in foster care.

Interestingly, when a conventional sequential algorithm is applied to this data set, variations of this consensus pattern overwhelm the results, because roughly half of the sequences in this data set followed the typical behavior approximately.

The rest of the sequences in this data set split into clusters of various sizes. One cluster formed around the 57 children who had short spells in foster care. The consensus pattern was $\langle(RPT)(INV, FC)(FC)(FC)\rangle$.

There were several consensus patterns from very small clusters with about 1% of the sequences. One such pattern of interest is shown below.

$$\langle(RPT)(INV, FC, T)(FC, T)\overbrace{(FC, HM)}^{8}(FC)(FC, HM)\rangle$$

where HM stands for Home Management Services and T stands for Transportation. There were 39 sequences

WS_1	$\langle(a:2, e:1):2$	$(h:1):1$	$(b:2):2$	$(d:2, e:1):2\rangle$	$:2$
S_4	$\langle(a)$		(bcg)	$(d)\rangle$	
WS_2	$\langle(a:3, e:1):3$	$(h:1):1$	$(b:3, c:1, g:1):3$	$(d:3, e:1):3\rangle$	$:3$

Figure 2: Sequences WS_1 and S_4 are aligned.

WS_2	$\langle(a:3, e:1):3$	$(h:1):1$	$(b:3, c:1, g:1):3$	$(d:3, e:1):3\rangle$		$:3$
S_5	$\langle$		(bci)	$(de)\rangle$		
WS_3	$\langle(a:3, e:1):3$	$(h:1):1$	$(b:4, c:2, g:1, i:1):4$	$(d:4, e:2):4\rangle$		$:4$
S_1	$\langle(ag)$	(f)	(bc)	(ae)	$(h)\rangle$	
WS_4	$\langle(a:4, e:1, g:1):4$	$(f:1, h:1):2$	$(b:5, c:3, g:1, i:1):5$	$(a:1, d:4, e:3):5$	$(h:1):1\rangle$	$:5$

Figure 3: The alignment of remaining sequences.

in the cluster. Our clients were interested in this pattern because foster care services and home management services were expected to be given as an "either/or" service, but not together to one child at the same time. Thus, this led us to go back to the original data to see if indeed many of the children received both services in the same month. Our investigation found that this was true, and lead our client to investigate this further in real practice. Was this a systematic data entry error or was there some components to Home Management Services (originally designed for those staying at home with their guardian) that were used in conjunction with Foster Care Services on a regular basis? Which counties were giving these services in this manner? Such an important investigation would not have been triggered without our analysis because no one ever suspected there was such a pattern. It is difficult to achieve the same results using the conventional sequential analysis methods because with *min_support* set to 20%, there is more than 100,000 sequential patterns and the users just cannot identify the needle from the straws.

6 Discussion and Conclusions

In this paper, we introduce ApproxMAP, a new approach to approximate sequential pattern mining. Its goal is to organize and summarize sequence of sets to uncover the underlying consensus patterns in the data. ApproxMAP uses clustering as a preprocessing step to group similar sequences, and then mines the underlying consensus patterns in each cluster directly through multiple alignment. A novel structure, weighted sequences, is proposed to summarize and compress the alignment information.

To the best of our knowledge, this is the first study on mining consensus patterns from sequence databases. It distinguishes itself from the previous studies in the following two aspects. First, it proposes the theme of approximate sequential pattern mining, which reduces number of patterns substantially and provides much more accurate and informative insights into sequential data. Second, it generalizes the multiple alignment techniques to handle sequences of itemsets. Mining sequences of itemsets extends the application domain substantially. The method is applicable to many interesting problems, such as business analysis, security, and complex bio-sequences analysis.

Our study illustrates that approximate sequential pattern mining can find general, useful, concise and understandable knowledge and thus is an interesting and promising direction.

References

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE 95*, pages 3–14, Taipei, Taiwan, Mar. 1995.

[2] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *SIGKDD*, pages 429–435, July 2002.

[3] R. J. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD*, pages 85–93, June 1998.

[4] J. Coggins. Dissimilarity measures for clustering strings. In *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison.* D. Snakoff, & J. Kruskal, (Eds.), pp 253-310. Addison-Wesley Pub. Co. MA. 1983.

[5] D. Gusfield. Algorithms on strings, trees, & sequences: Computer Science and Computational Biology. Cambridge Univ. Press, Cambridge, England. 1997.

[6] J. Han, J. Pei, et al. FreeSpan: Frequent pattern-projected sequential pattern mining. In *SIGKDD*, pages 355–359, Aug. 2000.

[7] H.C. Kum, J. Pei, W. Wang, and D. Duncan. ApproxMAP : Approximate Mining of Consensus Sequential Patterns. *Technical Report TR02-031*, UNC-CH, 2002.

[8] G. R. McPherson and S. DeStefano. Applied Ecology and Natural Resource Management. Cambridge University Press, Cambridge, England. 2002.

[9] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416, Jan. 1999.

[10] J. Pei, J. Han, et al. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, pages 215–224, April 2001.

[11] Sas Institute. Proc Modeclust. In SAS/STAT User Guide. Sas online Document. 2000

[12] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT*, pages 3–17, Mar. 1996.

[13] M. J. Zaki, S. Parthasarathy, et al. Parallel algorithm for discovery of association rules. *Data Mining and Knowledge Discovery*, 1:343–374, 1997.