Mining Cyclically Repeated Patterns

Ismail H. Toroslu[1] and Murat Kantarcioglu[2]

[1] University of Central Florida, Department of Computer Science,
Orlando, FL 32816-162362
toroslu@cs.ucf.edu
[2] Purdue University, Department of Computer Science,
West Lafayette, IN 47907-1398
kanmurat@cs.purdue.edu

Abstract. In sequential pattern mining, the support of the sequential pattern for the transaction database is defined only by the fraction of the customers supporting this sequence, which is known as the customer support. In this paper, a new parameter is introduced for each customer, called as repetition support, as an additional constraint to specify the minimum number of repetitions of the patterns by each customer. We call the patterns discovered using this technique as *cyclically repeated patterns*. The additional parameter makes the new mining technique more efficient and also helps discovering more useful patterns by reducing the number of patterns searched. Also, ordinary sequential pattern mining can be represented as a special case of the cyclically repeated pattern mining. In this paper, we introduce the concept of mining cyclically repeated patterns, we describe the related algorithms, and at the end of the paper we give some performance results.

1 Introduction

Several different forms of data mining problems have been investigated, and many different techniques have been developed. The original form of the sequential pattern mining problem started to get new attention [8,9,12], in addition to the early works on this problem [3,10]. The roots of the sequential pattern mining problem can be traced back to AI (as in [5]) and string processing (as in [11]). However, it got more attention from the database community after its redefinition in [3] as a database problem.

Sometimes, sequential pattern mining is treated under more general class of time-series analysis [6]. The time-series analysis problem had received more attention from the researchers. Some recent ones are presented in [1,4,7]. In this paper, we extend the very basic form of the sequential patterns to the cyclically repeated patterns, which is the generalization of the former one. The technique presented in [10] also generalizes the sequential pattern mining by considering temporal information on the transactions.

In order to define the concept of mining cyclically repeated patterns, a new parameter, called repetition support, is defined. In sequential pattern mining, there is only a single support constraint. Due to the additional constraint, in cyclically repeated pattern mining, the whole search process becomes much more efficient. In cyclically repeated pattern mining technique, since there will be less number of

Y. Kambayashi, W. Winiwarter, and M. Arikawa (Eds.): DaWaK 2001, LNCS 2114, pp. 83–92, 2001.

candidates compared to the sequential pattern mining, the memory requirement for the hash-table will significantly be reduced. Also, because of the additional constraint, there will be much less number of possible cyclically repeated patterns, and the algorithm will terminate more quickly. Data mining is basically used for decision support process. Therefore, introducing a new parameter will give the decision maker more flexibility in obtaining different results with two different types of supports. Also by producing less number of patterns it could be easier for the decision maker to interpret the results of the mining process.

We will also use the definitions similar to [2,3] in order to introduce the "cyclically repeated pattern mining" problem. Since the audience of this field is familiar with the customer transaction databases, throughout the paper we will also use customer transaction database as an example. Customer transaction database is a relation consisting of the following fields: customer id, transaction time, and items purchased. Itemset is any set of items, which can be purchased. Sequence is an ordered list of itemsets denoted as $[i_1, i_2, \ldots, i_n]$ where each i_p is an itemset. Cyclically repeated pattern is just a repeated sequence. Concatenation the sequence x k times can be represented as x^k. Thus, if x is a sequence $[i_1, i_2, \ldots, i_n]$, then x^k is $[i_1, i_2, \ldots, i_n, i_1, i_2, \ldots, i_n, \ldots, i_1, i_2, \ldots, i_n]$, where the subsequence $[i_1, i_2, \ldots, i_n]$ is repeated k times. When we have a sequence x^k, we call x (i.e., $[i_1, i_2, \ldots, i_n]$) as a cyclically repeated pattern with k repetitions, or a cyclically repeated pattern with repetition support k.

An example rule, which can be discovered from the customer transaction database, may be like: *It is likely that customers' purchase of "coke and chips" twice, is followed by a purchase of "milk and cookies*". If such a pattern is observed in sufficiently many number of customers, which is repeated in sufficiently many times by each customer (in which the patterns can start at any point of the pattern), then we can claim the existence of this pattern as cyclically repeated pattern.

A cyclically repeated pattern $[i_1, i_2, \ldots, i_n]$ represents all cyclically repeated patterns $[i_r, \ldots, i_n, i_1, i_2, \ldots, i_{r-1}]$, for all the values of r between 1 and n. We call all cyclically repeated patterns represented by $[i_1, i_2, \ldots, i_n]$ as the family of the cyclically repeated patterns. We choose the cyclically repeated pattern with the highest repetition support from its family to represent that family. Thus, each cyclically repeated pattern with length n represents a family of n patterns. If the representative of the family of the cyclically repeated patterns has k repetitions in a single customer's transactions, then all other patterns in the family have at least k-1 repetitions in that customer's transactions. Therefore, assuming all the cyclically repeated patterns in that family having repetition support k from that customer is a very reasonable estimation of the repetition support for those patterns. By using only the representative of the family we reduce the number of candidates in all the steps of the process.

The definition of the customer support is also adopted from [3]. In our technique, customer support of a pattern represents the fraction of customers with that pattern having the required repetition support. Therefore, the new cyclically repeated pattern mining technique can also be used to discover sequential patterns by just choosing the repetition support as one and discarding the "pattern families" concept and representing each candidate by themselves.

Even though there are also some new techniques proposed for sequential pattern mining (as in [8,12]), we have used the original algorithm of [3] since it can easily be adapted to the cyclically repeated pattern mining problem. Other algorithms might also be modified for this purpose. However, the main point of this paper is not just to show an efficiency of a specific algorithm, but the efficiency of the whole new

technique. Since in our approach the number of patterns discovered can be reduced using the repetition support, it works more efficiently than the sequential pattern mining technique in all steps. Also reducing the memory requirement contributes the execution time as well, since the memory requirements of these algorithms could be very large and main memory might not be sufficient and page swaps could be needed.

The rest of the paper is organized as follows: In section 2, the main steps of the algorithm are described. The main difference of cyclically repeated pattern mining technique from the sequential pattern mining is in the fourth phase, which we call repeated sequence phase in our algorithm, and it corresponds to the sequence phase of the sequential pattern mining technique of [3]. In section 3, we describe how the addition of the repetition support changes that phase. The details of this phase are given for processing single customer's transactions. Section 3 also describes how the algorithm works for the whole transaction database. In section 4, some performance results are given, and finally section 5 presents the conclusions.

2 General Cyclically Repeated Pattern Algorithm

In this section we define the main steps of our algorithm, which is actually based on the "a priori all" technique of [2,3]. We have extended the sequential pattern mining algorithm of [3]. Also we have made some modifications on the data structure in order to determine the repetition supports. Except the fourth phase, other phases of our technique are almost the same as the one in [3]. Below they are briefly defined. The details of these phases can be found in [3].

Phase 1: Sort Phase: The database is sorted using customer id as the major key and transaction time as the minor key.

Phase 2: Litemset Phase: The set of large item sets, satisfying the given supports, are determined. They are called as litemsets. These litemsets are mapped to contiguous integers after they are determined. The difference between this phase in our algorithm and the one in [3] is that, the litemsets in our technique should also satisfy the repetition supports for each customer. Extending the transformation phase of [3] with this additional support constraint is very simple. In order to increment the customer support count, the number of repetitions of the litemsets for each customer is determined. If the number of repetitions is greater than or equal to the required repetition support then the customer support is increased. As in [3], where litemsets correspond to large-1-sequences, the litemsets in our technique also correspond to large-1-cyclically-repeated-patterns.

Phase 3: Transformation Phase: This phase is the same as the one in [3]. The original database is transformed into a database of litemset identifiers by replacing each transaction by the set of all the litemsets in that transaction. Also, the transactions that do not appear in litemsets are removed from the database.

Phase 4: Repeated Sequence Phase: We used "a priori all" algorithm with some modification in this phase. The data structure has also been modified slightly. The details of this phase are given in the following sections.

Phase 5: Maximal Phase: Maximal cyclically repeated patterns are determined from the set of all large cyclically repeated patterns. This phase is also very similar to the one in [3]. Since each large cyclically repeated pattern represents a family of the patterns just a simple subsequence check among large cyclically repeated patterns is

not enough for the containment check. The subsequence check is extended in order to consider the subsequences that can be obtained by wrapping around the pattern.

3 Repeated Sequence Phase

The repeated sequence phase utilizes the "a priori" approach defined in [3]. This simply means that through the iterations, using the cyclically repeated patterns generated just in the previous iteration, larger cyclically repeated patterns are generated. At iteration *i*, possible candidates of that iteration, or *large-i-candidates* are generated using the *large-(i-1)-cyclically repeated-patterns* constructed in the previous iteration. These candidates are stored in a hash-tree structure in order to efficiently search each customer transaction sequence to determine their repetition supports and by combining those supports to determine their customer supports. Those candidates having at least the required minimum supports become *large-i-cyclically repeated-patterns*. In our technique there are some additions to the hash-tree data structure, and there are also some modifications in the candidate generation process and the hash-tree search process. We will use the following customer transactions sequence, which represents the list of the set of the litemsets obtained after the transformation phase:

[{1}, {2}, {2, 3, 4}, {3, 4}, {1}, {2}, {2, 3, 4}, {3, 4}, {1}]

Assuming the minimum required repetition support is 2, after the first two iterations the repetition supports of large-2-cyclically repeated-patterns satisfying the minimum support 2 will be obtained as follows:

Large-2-Cyclically repeated-Pattern	**Support**
[1, 2]	2
[1, 3]	2
[1, 4]	2
[2, 3]	2
[2, 4]	2
[3, 4]	2

At this iteration, the reverses of the some of the large-2-cyclically repeated-patterns should also be generated as large-2-cyclically repeated-patterns. However, we only keep one pattern from each family of patterns with the highest support representing that family. When the length of the patterns is 2, there will be only a pair of the form [a, b] and [b, a] in a family. The cyclically repeated patterns in the above table cover all the patterns with the required repetition support. In our example, all the reverse sequences either have the same or one less repetition supports than the ones in the above table.

When the number of distinct litemsets is *k*, the maximum number of possible candidates with length *n* is $k!/(k-n)!$. Only $k!/(n.(k-n)!)$ of them are needed to represent all of them. Therefore, we also reduce the number of the candidates (and therefore the size of the hash-tables) by the ratio of *n* (the length of the candidates) by keeping only one representative from each family of the patterns. In sequential pattern mining, all the candidates had to be represented in the hash-table. In this example, we have 4

distinct litemsets, and eight patterns can represent all possible 24 candidates that can be generated with length 3. No patterns from the families of [1, 4, 2] and [1, 3, 2] have any support above the required repetition support 2. Therefore we will obtain only 6 large-3-cyclically repeated-patterns after this iteration. Some of the patterns indirectly represented do not necessarily have to have the required repetition support, and they might have one less repetition support than their representatives. For example the pattern [3, 4, 2] represented by [2, 3, 4] has only repetition support 1. After this step the following repetition supports will be obtained for the patterns satisfying the repetition support:

Large-3-Cyclically repeated-Pattern	**Support**
[1, 2, 3]	2
[1, 2, 4]	2
[1, 3, 4]	2
[1, 4, 3]	2
[2, 3, 4]	2
[2, 4, 3]	2

We will describe the candidate generation process after the third iteration using the same example. This process is different than the one in [3] since we use only a representative of a family to represent all of the possible cyclically repeated patterns in that family. In order to generate the candidates with “length n+1” for the next iteration, n many “length n-1” subsequences of each large-n-cyclically repeated-patterns are searched for joining. A candidate with “length n+1” is obtained by joining two large-n-cyclically repeated-patterns found in the previous iteration (the iteration n) by using their common “length n-1” subsequences, and then by adding the remaining two litemsets of these two patterns. For the above example, from “length 3” to go to the next iteration with “length 4” we check the followings subsequences:

Large-3-Cyclically-Repeated-Pattern	**Subsequences**
[1, 2, 3]	[1, 2], [2, 3], [3, 1]
[1, 2, 4]	[1, 2], [2, 4], [4, 1]
[1, 3, 4]	[1, 3], [3, 4], [4, 1]
[1, 4, 3]	[1, 4], [4, 3], [3, 1]
[2, 3, 4]	[2, 3], [3, 4], [4, 2]
[2, 4, 3]	[2, 4], [4, 3], [3, 2]

Since each pattern represents a family of patterns (a pattern with length n represents n patterns) we should consider the whole family in order to determine the candidates of the next iteration. For example the patterns [1, 2, 3] and [2, 3, 4] can be used to generate candidates [2, 3, 1, 4] and [2, 3, 4, 1]. Here the pattern [2, 3, 1] is considered from the family represented by [1, 2, 3]. Among the subsequences of [2, 3, 1, 4], [1, 4, 2] does not have the required repetition support, which has been determined at the previous iteration. Therefore, it will be discarded from the candidate list of the next iteration.

During the generation of the new “length n+1” candidates for the next iteration, each new candidate should be checked if a previously generated candidate already represents it, and if so, that candidate is discarded. Among all these large-4-cyclically repeated-candidates, assuming they are generated in the order shown in the above table, after removing the ones represented by previously generated candidates, we will only have the following 6 candidates at this iteration: [1, 2, 3, 4], [1, 2, 4, 3], [3, 1, 4, 2], [4, 1, 3, 2], [3, 4, 2, 1], and [4, 3, 2, 1].

After this step, we should also remove those large-4-candidates having at least one large-3-subsequence that does not have enough support. Each large-n-cyclically repeated-candidate has n *large-(n-1)-subsequence*; some of them are obtained by wrapping around the pattern. Again, for the subsequences we have to look for, if their representatives have required supports (in our example the required support is 2). Actually all of the “length n-1” subsequences (total n of them) can be obtained, each time by deleting a different litemset from “length n” candidate, n times. The following table shows for our example, the supports of the candidates from their subsequences:

Candidate	**Subseqs with Support**	**Subseqs w/o Support**
[1, 2, 3, 4]	[1, 2, 3], [2, 3, 4], [3, 4, 1], [4, 1, 2]	
[1, 2, 4, 3]	[1, 2, 4], [2, 4, 3], [4, 3, 1], [3, 1, 2]	
[3, 1, 4, 2]	[3, 1, 4], [4, 2, 3], [2, 3, 1]	[1, 4, 2]
[4, 1, 3, 2]	[4, 1, 3], [3, 2, 4], [2, 4, 1]	[1, 3, 2]
[3, 4, 2, 1]	[3, 4, 2], [1, 3, 4]	[4, 2, 1], [2, 1, 3]
[4, 3, 2, 1]	[4, 3, 2], [1, 4, 3]	[3, 2, 1], [2, 1, 4]

Since only candidates [1, 2, 3, 4] and [1, 2, 4, 3] have all their subsequences supported in the previous iteration, they will be the only two candidates that will be considered in this iteration. For our example customer sequence we will get repeated support 2 for both candidates making them as large-4-cyclically repeated-patterns.

Next, we will use the following sample database in order to explain the remaining details of the repeated sequence phase.

Customer 1: [{1}, {1}, {3}, {1}, {3}, {2}, {4}, {2, 3}, {1}, {3, 4}]
Customer 2: [{3, 4}, {1}, {1, 2, 3, 5}, {2, 4}, {1, 2, 5}]
Customer 3: [{2} {1, 4}, {3, 4}, {2, 3}, {1}, {4}, {2, 4}, {1, 2, 3, 5}, {1}, {1, 2, 3, 4, 5}]
Customer 4: [{4}, {2, 3}, {1, 4}, {1, 2, 3, 5}, {1, 2, 4, 5}, {2, 3, 4}, {1, 2, 4, 5}, {4}]

On this database the following support values are used:

- Repetition support = 2 (which means a pattern must be repeated at least twice by a customer in order to be considered as supported).
- Customer support = 3 (which means there must be at least 3 different customers for the pattern satisfying the given repetition support).

Basically, we will use the customer support, as in the ordinary sequential pattern mining for eliminating patterns without sufficient supports. However, in determining the customer support, we also take repetition support of the pattern into account for each customer. Simply when we process a customer transaction we increment the customer support count only for the patterns that also satisfy the repetition support constraint.

Each one of the integers in this database represents a litemset with sufficient customer (and therefore repetition) support. In the repeated sequence phase of our algorithm, similar to the ordinary sequential mining techniques, maximal large cyclically repeated patterns are determined. In order to do this, at each iteration step, larger item sets should be generated with the required supports. In this section we will also use the term *support* to represent customer supports satisfying the repetition support conditions. Initially, these litemsets have the following supports:

Litemset	**Support**
1	4
2	4
3	4
4	4
5	3

In the next iteration large-2-cyclically repeated-patterns satisfying the both support conditions are determined. Note that many patterns are eliminated since they do not have sufficient customer supports with the required repetition supports. For example the pattern [1, 2] does not have enough repetition support for customer 1 and customer 2 (which is just 1), although the same pattern have repetition support 3 for the customer 3 and customer 4. Although the pattern [1, 2] has a repetition support 2 for customer 4, we actually say it has a repetition support 3, by using the repetition support of the pattern [2, 1] which is the representative of the family pattern since it has the highest support for the family of [1, 2] and [2, 1].

When only the representatives of the pattern families are used, we will have the following large-2-cyclically repeated-patterns with required supports:

Large-2-cyclically repeated-pattern	**Support**
[3, 1]	4
[4, 1]	3
[4, 2]	3

After all the patterns with 2 litemsets having sufficient supports are determined, in the next step, only one cyclically repeated pattern with 3 litemsets will be determined. That will be [3,1,4] with support 3. These iterations continue until no cyclically repeated pattern is found for some length. In our example since there is no cyclically repeated pattern with length 4 having required supports, the algorithm terminates after the third iteration.

There is only a small difference in the new hash-tree-insert algorithm from the one in [3]. In order to determine the repetition supports of the candidates we have to keep additional information for each candidate, namely *the count of the candidate* and *the ending transaction number* of its last occurrence in a currently processed customer transaction.

The hash-tree search algorithm has more modifications than the one in [3], which is used to find only sequential patterns. In the new hash-tree-search algorithm whenever a candidate is detected in a customer sequence, its repetition support count has to be incremented and its end transaction number of its last occurrence up to that transaction in a customer sequence has to be recorded The modifications in the hash-

tree search algorithm in order to count all the occurrences of the candidates are as follows: The starting transaction number of the partially generated candidate that is being searched is also passed as a parameter to the search algorithm. Whenever the searched pattern is found as a candidate in the hash-tree, its last occurrence ending transaction number is compared to the starting transaction number of the currently being searched occurrence to determine whether the currently being searched occurrence had started after the last occurrence had ended. Only if that is the case, the counter of the candidate is incremented, and the ending transaction number of the candidate is updated as the current transaction number.

4 Performance Results

We have used one of the datasets of [3] in our performance evaluations (the dataset C10-T2.5-S4-I1.25 of [3] with 25000 customers and with customer support as %1). In this dataset there are only a few and very short cyclically repeated patterns (with length 2 only), and there is no cyclically repeated pattern with repetition support 4 or 5. Since there were not many cyclically repeated patterns in this dataset we have created a new datasets from C10-T2.5-S4-I1.25 by using the first 5000 customers and repeating its transactions several times. We present the results of one of these files with 4 repetitions in Figure 1. We called the new dataset as C10-T2.5-S4-I1.5-4. The rest of the parameters in these datasets have not been changed. Since the other phases in our technique and the one in [3] are the same, in the Figures 1, we present the evaluation times of the repeated sequence phases only after normalizing the execution times according to repetition support 1 as 100. The results of other test cases were also very similar after the normalizations.

As it can be seen from this figure, there is a sharp decrease in the execution times of the repeated sequence phase of the algorithm when the repetition support increases. Even for the repetition support 2 there is a significant decrease in all the cases. Although we did not obtain the results for the ordinary sequential pattern search algorithm, it is clear that it will take more time than the cyclically repeated pattern algorithm even with repetition support 1. In cyclically repeated pattern search technique there are always fewer candidates than the sequential pattern search technique, because of representing pattern families with only one pattern.

These results can be interpreted in a broad way. They do not only show the different behaviors of the execution times of two algorithms, but in general the overall contribution of the “cyclically repeated pattern mining” approach over “sequential pattern mining”. Since the main reason for this improvement is due to the decrease in the number of candidates searched, similar kind of improvement should be expected from other kind of implementations of these techniques (i.e. by modifying other sequential pattern mining algorithms for the cyclically repeated pattern mining).

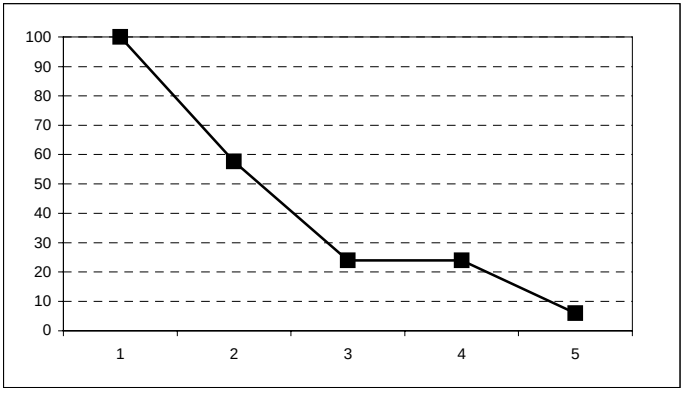

Fig. 1. Execution times of the repeated sequence phase, normalized according to repetition support 1, for the dataset C10-T2.5-S4-I1.5-4 with repetition supports 1 to 5 and customer supports 1%.

5 Conclusion

In this paper, we have presented an extension to the sequential data mining technique. We have defined the concept of cyclically repeated pattern mining by introducing a new support criterion. We have modified the data structures and the algorithms presented in [3] order to discover cyclically repeated patterns. We have also introduced the concept of the pattern families as a natural extension to the cyclically repeated pattern mining. The popular customer transaction database is used in order to describe the new technique.

Our performance results show that, searching for cyclically repeated patterns with higher repetition supports significantly decrease the execution time. In most cases, cyclically repeated pattern mining approach will discover less number of patterns than the sequential pattern mining, because, the repetition support introduces an extra constraint for the patterns, and also representing the family of patterns with a single pattern reduces the possible number of patterns. This affects positively all the steps of the mining process by decreasing the number of candidates and by reducing the sizes of hash trees. As a result, in cyclically repeated pattern mining technique the memory requirement is also reduced.

References

1. R. Agrawal, C. Faloutsos, A. Swami: “Efficient similarity search in sequence databases”, *Proc. of the 4th Int’l Conference on Foundations of Data Organization and Algorithms*, Chicago, Oct. 1993, Also in Lecture Notes in Computer Science 730, Springer Verlag, pp. 69-84, 1993.

2. R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", *Proc. of the ACM SIGMOD Conference on Management of Data*, pp. 207-216, Washington D.C., May 1993.
3. R. Agrawal, R. Srikant, "Mining sequential patterns", *Proc. of the Int'l Conference on Data Engineering (ICDE)*, pp. 3-14, Taipei, Taiwan, March 1995.
4. C. Bettini, X. S. Wang, S. Jajodia, "Mining temporal relationships with multiple granularities in time sequences", *Data Engineering Bulletin*, Vol. 21, pp. 32-38, 1998.
5. T. G. Dietterich, R.S. Michalski, "Discovering patterns in sequence of events", *Artificial Intelligence*, vol. 25, pp. 187-232, 1985.
6. J. Han, "Data mining", *Encyclopedia of Distributed Computing*, Eds. J. Urban and P. Dasgupta, Kluwer Academic Publishers, 1999.
7. J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M-C. Hsu, Freespan: Frequent pattern-projected sequential pattern mining", *In Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD'00)*, pp. 355-359, Boston, MA, Aug. 2000.
8. J. Han, J. Pei, Y. Yin, "Mining frequent patterns without candidate generation", *In Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, pp. 1-12, Dallas, TX, May 2000.
9. M. Garofolakis, R. Rastogi, K. Shim, "Spirit: Sequential pattern mining with regular expression constraints", *In Proc. 1999 Int. Conf. Very Large Data Bases (VLDB'99)*, pp. 223-224, Edinburgh, UK, Sept. 1999.
10. R. Srikant, R. Agrawal, "Mining sequential patterns: generalizations and performance improvements". pp. 3-17, *Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT)*, Avignon, France, March 1996.
11. S. Wu, U. Manber, "Fast text searching allowing errors", Communications of the ACM, vol. 35, no. 10, pp. 83-91, Oct. 1992.
12. M. J. Zaki, "Efficient Enumeration of Frequent Sequences", *7th International Conference on Information and Knowledge Management*, pp 68-75, Washington DC, November 1998.