The PrefixSpan Approach to Mining Sequential Patterns by Pattern Growth

Sujit R Wakchaure Dr. Rajeev G Vishwakarma
Department of Computer Science and Engineering,
Dr. A. P. J. Abdul Kalam University, Indore (M.P.) - 452016, India

Abstract:
The PrefixSpan approach to mining sequential patterns is a pattern growth method in which a pattern is extended one item at a time. The data is first scanned, and frequent patterns are identified. Then, the identified patterns are extended to discover longer patterns. The PrefixSpan algorithm uses a prefix tree to store the candidate patterns, which are then extended by adding one item at a time. The process is repeated until no more patterns can be found. The PrefixSpan approach is useful in uncovering complex patterns in large datasets, such as those found in biological and logistic processes. This approach is also suitable for streaming data, as the patterns can be updated in real time. Additionally, the prefix tree can be used to efficiently store and access the patterns and their extensions. Sequential pattern mining is a significant data mining problem with a wide range of applications. It is, however, a difficult problem because mining may have to generate or examine a combinatorially explosive number of intermediate subsequences. To reduce the number of candidates to be examined, most previously developed sequential pattern mining methods, such as GSP, employ a candidate generation-and-test approach. However, this method may be inefficient when mining large sequence databases with many patterns and/or long patterns. In this paper, we propose a sequential pattern-growth approach based on projections for efficient mining of sequential patterns.

Keywords: sequential pattern, frequent pattern, sequence database, scalability, performance analysis, transaction database.
DOI: 10.24297/j.cims.2023.4.2

1. Introduction

Sequential pattern mining, which identifies frequent subsequences as patterns in a sequence database, is a significant data mining problem with numerous applications, such as the analysis of customer purchase patterns or Web access patterns, the analysis of sequencing or time-related processes such as scientific experiments, natural disasters, and disease treatments, the analysis of DNA sequences, and so on.

Agrawal and Srikant introduced the sequential pattern mining problem in [2]: Given a set of sequences, where each sequence is made up of a list of elements, and each element is made up of a set of items, and a user-specified min support threshold, the goal of sequential pattern mining is to find all frequent subsequences, that is, subsequences whose occurrence frequency in the set of sequences is greater than min support.

A typical a priori-like sequential pattern mining method, such as GSP [23], employs a multiple-pass, candidate generation-and-test approach, as shown below: The first scan finds all of the frequent items that comprise the set of single item frequent sequences. Each subsequent pass begins with a seed set of sequential patterns, which is the set of sequential patterns discovered in the previous pass. Based on the a priori principle, this seed set is used to generate new potential patterns known as candidate sequences. Each candidate sequence contains one more item than a seed sequential pattern, where each element in the pattern may contain one or more items. The length of a sequence is the number of items in the sequence. As a result, all of the candidate sequences in a pass will be the same length. In one pass, the database is scanned to find support for each candidate sequence. The set of newly discovered sequential patterns consists of all candidates in the database with support greater than min support. This set becomes the seed set for the next pass. When no new sequential pattern is found in a pass, or when no candidate sequence can be generated, the algorithm terminates.

Although it reduces search space, the a priori-like sequential pattern mining method has three nontrivial, inherent costs that are independent of detailed implementation techniques.
sequence generation, test, and support counting is inherent to the a priori-based method, no matter what technique is applied to optimize its detailed implementation.

TABLE 1 A Sequence Database

Sequence_id	Sequence
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle(ad)c(bc)(ae)\rangle$
30	$\langle(ef)(ab)(df)cb\rangle$
40	$\langle eg(af)cbc\rangle$

2. Problem Definition And The Gsp Algorithm

In this section, the problem of sequential pattern mining is defined, and the most representative a priori-based sequen- tial pattern mining method, GSP [23], is illustrated using an example.

Problem Definition

Let I ¼ fi1; i2; … ; ingbe a set of all items. An itemset is a subset of items. A sequence is an ordered list of itemsets. A sequence s is denoted by hs1s2 ··· sli, where sj is an itemset. sj is also called an element of the sequence, and denoted as ðx1x2 ··· xmÞ, where xk is an item. For brevity, the brackets are omitted if an element has only one item, i.e., element ðxÞ 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 2 ¼ ha1a2 ··· ani is called a subsequence of another sequence β ¼ hb1b2 ··· bmi and β a supersequence of 2, denoted as 2 v β, if there exist integers 1 ≤ j1 < j2 < ··· < jn ≤ m such that a1 ⊆ bj1 ; a2 ⊆ bj2 ; … ; an ⊆ bjn .

A sequence database S is a set of tuples hsid; si, where sid is a sequence_id and s a sequence. A tuple hsid; si issaid to contain a sequence 2, if 2 is a subsequence of s. The support of a sequence 2 in a sequence database S is the number of tuples in the database containing 2, i.e., supportSð2Þ ¼ j fhsid; sijðhsid; si 2 SÞ ^ ð2 v sÞgj :

It can be denoted as supportð2Þ if the sequence database is clear from the context. Given a positive integer min_support as the support threshold, a sequence 2 is called a sequentialpattern in sequence database S if supportSð2Þ ≥ min support. A sequential pattern with length l is called an l-pattern.

Example 1 (Running Example). Let our running sequence database be S given in Table 1 and min support ¼ 2. The set of items in the database is fa; b; c; d; e; f; gg.

A sequence haðabcÞðacÞdðcfÞi has five elements: ðaÞ, ðabcÞ, ðacÞ, ðdÞ, and ðcfÞ, where items a and c appear more than once, respectively, in different elements. It is a 9-sequence since there are nine instances appearing in thatsequence. Item a happens three times in this sequence, soit contributes 3 to the length of the sequence. However, the whole sequence haðabcÞðacÞdðcfÞi contributes only 1 to the support of hai. Also, sequence haðbcÞdfi is a subsequence of haðabcÞðacÞdðcfÞi. Since both sequences 10 and 30 contain subsequence s ¼ hðabÞci, s is a sequential pattern of length 3 (i.e., 3-pattern).

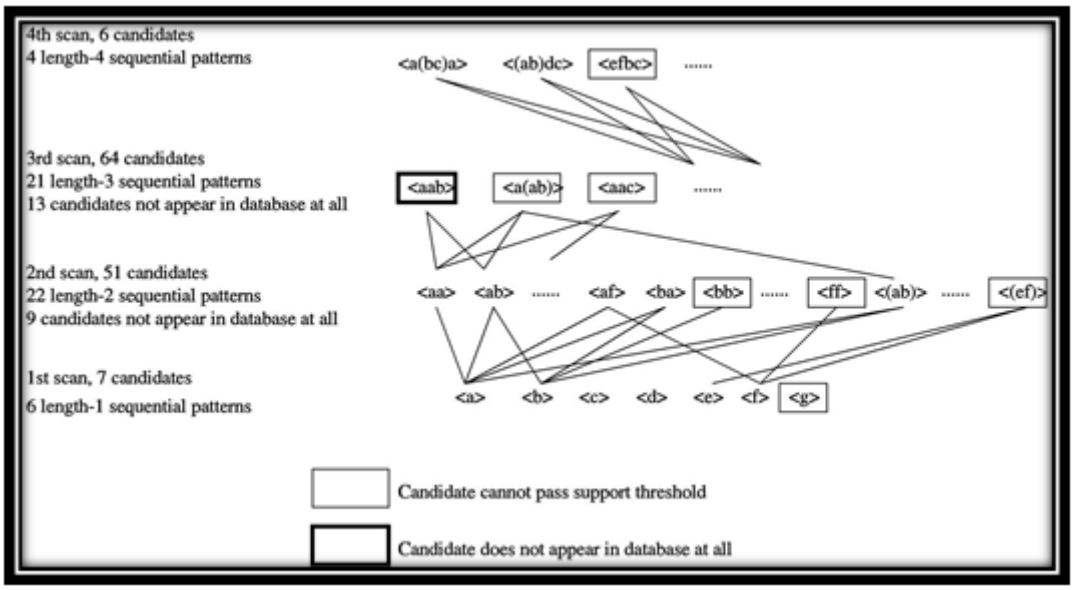


Fig. 1. Candidates and sequential patterns in GSP.

GSP, though benefits from the a priori pruning, still generates a large number of candidates. In this example, 6 length-1 sequential patterns generate 51 length-2 candidates, 22 length-2 sequential patterns generate 64 length-3 candidates, etc. Some candidates generated by GSP may not appear in the database at all. For example, 13 out of 64 length-3 candidates do not appear in the database.

3. Mining Sequential Patterns By Pattern Growth

The GSP algorithm, as discussed in Section 1 and Example 2, has similar strengths and weaknesses to the a priori method. A frequent pattern growth method known as FP-growth [9] has been developed for efficient mining of frequent patterns without candidate generation. To achieve high performance, the method employs a data structure known as an FP-tree to store compressed frequent patterns in a transaction database and recursively mines the projected conditional FP-trees.

Can we mine sequential patterns by extension of the FP-tree structure? Unfortunately, the answer cannot be so optimistic because it is easy to explore the sharing among a set of unordered items, but it is difficult to explore the sharing of common data structures among a set of ordered items.

For example, a set of frequent itemsets fabc; cbad; ebadc; cadbg share the same tree branch habcdei in the FP-tree. However, if they were a set of sequences, there is no common prefix

subtree structure that can be shared among them because one cannot change the order of items to form sharable prefix subsequences.

After repeatedly scanning the entire database and generating and testing large sets of candidate sequences, a sequence database can be recursively projected into a set of smaller databases associated with the set of patterns mined thus far, and then mine locally frequent patterns in each projected database.

In this section, we outline a projection-based sequential pattern mining method called FreeSpan [8], followed by a systematic introduction to an improved method called PrefixSpan [19].

TABLE 2: Projected Databases and Sequential Patterns

prefix	projected (suffix) database	sequential patterns
$\langle a \rangle$	$\langle (abc)(ac)d(cf) \rangle$, $\langle (\_d)c(bc)(ae) \rangle$, $\langle (\_b)(df)cb \rangle$, $\langle (\_f)cbc \rangle$	$\langle a \rangle$, $\langle aa \rangle$, $\langle ab \rangle$, $\langle a(bc) \rangle$, $\langle a(bc)a \rangle$, $\langle aba \rangle$, $\langle abc \rangle$, $\langle (ab) \rangle$, $\langle (ab)c \rangle$, $\langle (ab)d \rangle$, $\langle (ab)f \rangle$, $\langle (ab)dc \rangle$, $\langle ac \rangle$, $\langle aca \rangle$, $\langle acb \rangle$, $\langle acc \rangle$, $\langle ad \rangle$, $\langle adc \rangle$, $\langle af \rangle$
$\langle b \rangle$	$\langle (\_c)(ac)d(cf) \rangle$, $\langle (\_c)(ae) \rangle$, $\langle (df)cb \rangle$, $\langle c \rangle$	$\langle b \rangle$, $\langle ba \rangle$, $\langle bc \rangle$, $\langle (bc) \rangle$, $\langle (bc)a \rangle$, $\langle bd \rangle$, $\langle bdc \rangle$, $\langle bf \rangle$
$\langle c \rangle$	$\langle (ac)d(cf) \rangle$, $\langle (bc)(ae) \rangle$, $\langle b \rangle$, $\langle bc \rangle$	$\langle c \rangle$, $\langle ca \rangle$, $\langle cb \rangle$, $\langle cc \rangle$
$\langle d \rangle$	$\langle (cf) \rangle$, $\langle c(bc)(ae) \rangle$, $\langle (\_f)cb \rangle$	$\langle d \rangle$, $\langle db \rangle$, $\langle dc \rangle$, $\langle dcb \rangle$
$\langle e \rangle$	$\langle (\_f)(ab)(df)cb \rangle$, $\langle (af)cbc \rangle$	$\langle e \rangle$, $\langle ea \rangle$, $\langle eab \rangle$, $\langle eac \rangle$, $\langle eacb \rangle$, $\langle eb \rangle$, $\langle ebc \rangle$, $\langle ec \rangle$, $\langle ecb \rangle$, $\langle ef \rangle$, $\langle efb \rangle$, $\langle efc \rangle$, $\langle efcb \rangle$.
$\langle f \rangle$	$\langle (ab)(df)cb \rangle$, $\langle cbc \rangle$	$\langle f \rangle$, $\langle fb \rangle$, $\langle fbc \rangle$, $\langle fc \rangle$, $\langle fcb \rangle$

Divide search space. The complete set of sequen- tial patterns can be partitioned into the following six subsets according to the six prefixes:

1) the ones with prefix hai,

2) the ones with prefix hbi; ... ; and

3) the ones with prefix hfi.

Find subsets of sequential patterns. The subsets of sequential patterns can be mined by construct- ing the corresponding set of projected databases and mining each recursively. The projected databases as well as sequential patterns found in them are listed in Table 2, while the mining process is explained as follows:

4. Experimental Results And Performance Analysis

Since GSP [23] and SPADE [29] are the two most influential sequential pattern mining algorithms, we conduct an extensive performance study to compare PrefixSpan with them. In this section, we first report our experimental results on the performance of PrefixSpan in comparison with GSP and SPADE and, then, present an indepth analysis on why PrefixSpan outperforms the other algorithms.

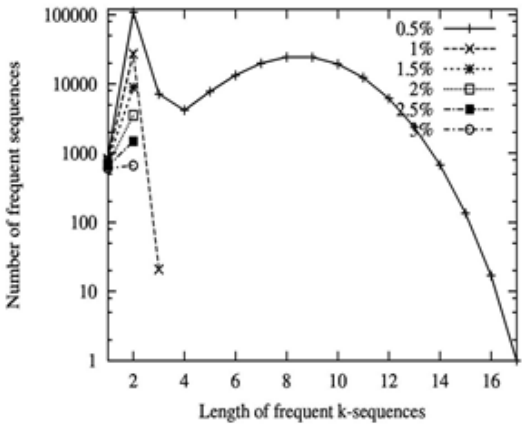
Experimental Results

To evaluate the effectiveness and efficiency of the PrefixSpan algorithm, we performed an extensive performance study of four algorithms: PrefixSpan, FreeSpan, GSP, and SPADE, on both real and synthetic data sets, with various kinds of sizes and data distributions.

All experiments were conducted on a 750 MHz AMD PC with 512 megabytes main memory, running Microsoft Windows 2000 Server. Three algorithms, GSP, FreeSpan, and PrefixSpan, were implemented by us using Microsoft Visual C++ 6.0. The implementation of the fourth algorithm, SPADE, is obtained directly from the author of the algorithm [29]. Detailed algorithm implementation is described as follows:

1. GSP. The GSP algorithm is implemented according to the description in [23].
2. SPADE. SPADE is tested with the implementation provided by the algorithm inventor [29].
3. FreeSpan. FreeSpan is implemented according to the algorithm described in [8].
4. PrefixSpan. PrefixSpan is implemented as described in this paper,3 with pseudoprojection turned on in most cases. Only in the case when testing the role of pseudoprojection, two options are adopted: one with the pseudoprojection function turned on and the other with it turned off.

For the data sets used in our performance study, we use two kinds of data sets: one real data set and a group of synthetic data sets.



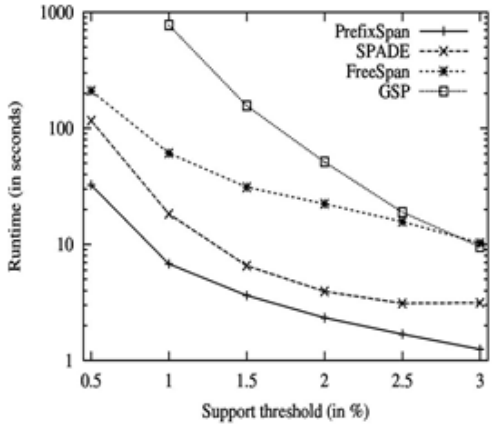


Fig. 2. Distribution of frequent sequences of data set.Fig. 3. Performance of the four algorithms on data set

SCOPE AND DISCUSSIONS

Mining sequential patterns, as opposed to mining (unordered) frequent patterns, is a step towards mining more sophisticated frequent patterns in large databases. Following the successful development of a pattern-growth-based sequential pattern mining method, such as PrefixSpan, it is interesting to investigate how such a method can be extended to handle more complex cases. This method is easily extended to mining multidimensional, multilevel sequential patterns [21]. This section will go over constraint-based mining of sequential patterns as well as a few research problems.

Mining Sequential Patterns Using Constraints Instead of finding all possible sequential patterns in a database, a user may prefer to enforce certain constraints to find desired patterns in many sequential pattern mining applications. Constraint-based mining refers to the mining process that incorporates user-specified constraints to reduce search space and derive only user-interested patterns.

A common single item from the set of fb; c; dg should be kept in the hai-projected database. Second, the remaining mining can begin with the suffix, which is essentially "Suffix-Span," a symmetric algorithm to PrefixSpan that grows suffixes from the end of the sequence forward. The growth should correspond to the suffix constraint "hfbbjbcdjddgi." To find all the remaining sequential patterns in the projected databases that match these suffixes, grow sequential patterns in either a prefix or suffix-expansion manner.

Problems facing during Research:

Although the sequential pattern growth approach proposed in this paper is efficient and scalable, there are still some difficult research issues in sequential pattern mining, particularly for certain large-scale applications. Here, we highlight a few issues that require further investigation.

Mining Closed and Maximal Sequential Patterns

As shown in Section 1, a frequent long sequence contains a combinatorial number of frequent subsequences. There are 2100 — 1 nonempty subsequences for a sequential pattern of length 100. In such cases, mining the entire set of patterns, regardless of method, is prohibitively expensive.

Similar to mining closed and maximal frequent patterns in transaction databases [17], [3], which mines only the longest frequent patterns (in the case of max-pattern mining) or the longest one with the same support (in the case of closed-pattern mining), it is also desirable to mine only (frequent) maximal or closed sequential patterns, where a sequence s is maximal if there is no frequent super sequence of s, and a sequence s is closed if there exists no frequent super sequence

Mining Approximate Sequential Patterns

We assumed in this study that all of the sequential patterns to be mined are exact matching patterns. Many applications require approximate matches in practise, such as DNA sequence analysis, which allows for limited insertions, deletions, and mutations in their sequential patterns. The development of efficient and scalable algorithms for mining approximate sequential patterns is both a difficult and practical endeavour. A recent study [28] on mining long sequential patterns in a noisy environment is an example of this.

5. Conclusions

We conducted a systematic study on sequential pattern mining in large databases and developed a pattern-growth approach for efficient and scalable sequential pattern mining.

Instead of refining the a priori-like, candidate generation-and-test approach, such as GSP [23], we promote a divide-and-conquer approach called pattern-growth [9], an efficient pattern-growth algorithm for mining frequent patterns without candidate generation. This paper proposes and investigates PrefixSpan, an efficient pattern-growth method. PrefixSpan recursively divides a

sequence database into smaller projected sequence databases and grows sequential patterns in each projected database by exploring only locally frequent fragments. It mines the entire set of sequential patterns and significantly reduces the effort required to generate candidate subsequences. Because PrefixSpan investigates ordered growth through prefix-ordered expansion, it produces fewer "growth points" and smaller projected databases than our previous proposed pattern-growth algorithm, FreeSpan. In addition, a pseudoprojection technique for PrefixSpan is proposed to reduce the number of physical projected databases generated.

The implications of this method, in our opinion, go far beyond yet another efficient sequential pattern mining algorithm. It demonstrates the strength of the pattern-growth mining methodology by achieving high performance in both frequent-pattern mining and sequential pattern mining. Furthermore, the methodology can be extended to mining multilevel, multidimensional sequential patterns, mining sequential patterns with user-specified constraints, and so on. As a result, it is a promising approach for applications that rely on the discovery of frequent and/or sequential patterns. Many intriguing issues need to be investigated further, such as mining closed and maximal sequential patterns, mining approximate sequential patterns, and extending the method to mining structured patterns. Future research should focus on the development of specialised sequential pattern mining methods for specific applications, such as DNA sequence mining that may admit flaws, such as allowing insertions, deletions, and mutations in DNA sequences, and handling industry/engineering sequential process analysis.

References

1. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. 1994 Int' l Conf. Very Large Data Bases (VLDB ' 94), pp. 487-499, Sept. 1994.
2. R. Agrawal and R. Srikant, "Mining Sequential Patterns," Proc. 1995 Int' l Conf. Data Eng. (ICDE ' 95), pp. 3-14, Mar. 1995.
3. R.J. Bayardo, "Efficiently Mining Long Patterns from Databases," Proc. 1998 ACM-SIGMOD Int' l Conf. Management of Data (SIGMOD ' 98), pp. 85-93, June 1998
4. R.J. Bayardo, R. Agrawal, and D. Gunopulos, "Constraint-Based Rule Mining on Large, Dense Data Sets," Proc. 1999 Int' l Conf. Data Eng. (ICDE ' 99), pp. 188-197, Apr. 1999.
5. C. Bettini, X.S. Wang, and S. Jajodia, "Mining Temporal Relation- ships with Multiple Granularities in Time Sequences," Data Eng. Bull., vol. 21, pp. 32-38, 1998.
6. S. Guha, R. Rastogi, and K. Shim, "Rock: A Robust Clustering Algorithm for Categorical Attributes," Proc. 1999 Int' l Conf. Data Eng. (ICDE ' 99), pp. 512-521, Mar. 1999.

7. J. Han, G. Dong, and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database," Proc. 1999 Int' l Conf. Data Eng. (ICDE ' 99), pp. 106-115, Apr. 1999.
8. J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining," Proc. 2000 ACM SIGKDD Int' l Conf. Knowledge Discovery in Databases (KDD ' 00), pp. 355-359, Aug. 2000
9. J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proc. 2000 ACM-SIGMOD Int' l Conf. Management of Data (SIGMOD ' 00), pp. 1-12, May 2000.
10. R. Kohavi, C. Brodley, B. Frasca, L. Mason, and Z. Zheng, "KDD- Cup 2000 Organizers' Report: Peeling the Onion," Proc. SIGKDD Explorations, vol. 2, pp. 86-98, 2000.
11. M. Kuramochi and G. Karypis, "Frequent Subgraph Discovery," Proc. 2001 Int' l Conf. Data Mining (ICDM ' 01), pp. 313-320, Nov. 2001.
12. H. Lu, J. Han, and L. Feng, "Stock Movement and n-Dimensional Inter-Transaction Association Rules," Proc. 1998 SIGMOD Work- shop Research Issues on Data Mining and Knowledge Discovery (DMKD ' 98), vol. 12, pp. 1-7, June 1998.
13. H. Mannila, H Toivonen, and A.I. Verkamo, "Discovery of Frequent Episodes in Event Sequences," Data Mining and Knowl- edge Discovery, vol. 1, pp. 259-289, 1997.
14. F. Masseglia, F. Cathala, and P. Poncelet, "The PSP Approach for Mining Sequential Patterns," Proc. 1998 European Symp. Principle of Data Mining and Knowledge Discovery (PKDD ' 98), pp. 176-184, Sept. 1998.
15. R. Ng, L.V.S. Lakshmanan, J. Han, and A. Pang, "Exploratory Mining and Pruning Optimizations of Constrained Associations Rules," Proc. 1998 ACM-SIGMOD Int' l Conf. Management of Data (SIGMOD ' 98), pp. 13-24, June 1998.
16. B. O¨ zden, S. Ramaswamy, and A. Silberschatz, "Cyclic Associa- tion Rules," Proc. 1998 Int' l Conf. Data Eng. (ICDE ' 98), pp. 412-421, Feb. 1998.
17. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering Frequent Closed Itemsets for Association Rules," Proc. Seventh Int' l Conf. Database Theory (ICDT ' 99), pp. 398-416, Jan. 1999.
18. J. Pei, J. Han, and L.V.S. Lakshmanan, "Mining Frequent Itemsets with Convertible Constraints," Proc. 2001 Int' l Conf. Data Eng. (ICDE ' 01), pp. 433-332, Apr. 2001.
19. 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," Proc. 2001 Int' l Conf. Data Eng. (ICDE ' 01), pp. 215-224, Apr. 2001.

20. J. Pei, J. Han, and W. Wang, "Constraint-Based Sequential Pattern Mining in Large Databases," Proc. 2002 Int' l Conf. Information and Knowledge Management (CIKM ' 02), pp. 18-25, Nov. 2002.
21. H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal, "Multi- Dimensional Sequential Pattern Mining," Proc. 2001 Int' l Conf. Information and Knowledge Management (CIKM ' 01), pp. 81-88, Nov. 2001.
22. S. Ramaswamy, S. Mahajan, and A. Silberschatz, "On the Discovery of Interesting Patterns in Association Rules," Proc. 1998 Int' l Conf. Very Large Data Bases (VLDB ' 98), pp. 368-379, Aug. 1998.
23. R. Srikant and R. Agrawal, "Mining Sequential Patterns: General- izations and Performance Improvements," Proc. Fifth Int' l Conf. Extending Database Technology (EDBT ' 96), pp. 3-17, Mar. 1996.
24. J. Wang, G. Chirn, T. Marr, B. Shapiro, D. Shasha, and K. Zhang, "Combinatiorial Pattern Discovery for Scientific Data: Some Preliminary Results," Proc. 1994 ACM-SIGMOD Int' l Conf. Management of Data (SIGMOD ' 94), pp. 115-125, May 1994.
25. Dr.U D Butkar and et.al "Modelling and Simulation of symmetric planar manipulator Using Hybrid Integrated Manufacturing." Computer Integrated Manufacturing Systems, 29(1), 464–476.
26. X. Yan and J. Han, "CloseGraph: Mining Closed Frequent Graph Patterns," Proc. 2003 ACM SIGKDD Int' l Conf. Knowledge Discovery and Data Mining (KDD ' 03), Aug. 2003.
27. X. Yan, J. Han, and R. Afshar, "CloSpan: Mining Closed Sequential Patterns in Large Datasets," Proc. 2003 SIAM Int' l Conf. Data Mining (SDM ' 03), pp. 166-177, May 2003.
28. J. Yang, W. Wang, P.S. Yu, and J. Han, "Mining Long Sequential Patterns in a Noisy Environment," Proc. 2002 ACM-SIGMOD Int' l Conf. Management of Data (SIGMOD ' 02), pp. 406-417, June 2002.
29. M. Zaki, "SPADE: An Efficient Algorithm for Mining Frequent Sequences," Machine Learning, vol. 40, pp. 31-60, 2001.
30. M.J. Zaki, "Efficient Enumeration of Frequent Sequences," Proc. Seventh Int' l Conf. Information and Knowledge Management (CIKM ' 98), pp. 68-75, Nov. 1998.
31. M.J. Zaki, "Efficiently Mining Frequent Trees in a Forest," Proc. 2002 ACM SIGKDD Int' l Conf. Knowledge Discovery in Databases (KDD ' 02), pp. 71-80, July 2002.