行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※※
※ ※
※ 互動式勘測大型資料庫中循序樣式之研究 ※
※ ※
※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：■個別型計畫 □整合型計畫
計畫編號：NSC 90-2213-E-009-078
執行期間： 90 年 8 月 1 日至 91 年 7 月 31 日

主持人：李素瑛 國立交通大學資訊工程學系
計畫參與人員：林明言 國立交通大學資訊工程學系

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

執行單位：國立交通大學資訊工程學系

中 華 民 國 91 年 10 月 30

行政院國家科學委員會專題研究計畫成果報告

互動式勘測大型資料庫中循序樣式之研究

Interactive Discovery of Sequential Patterns in Large Databases

計畫編號：NSC 90-2213-E-009-078

執行期限：90 年 8 月 1 日至 91 年 7 月 31 日

主持人：李素瑛　　國立交通大學資訊工程學系

計畫參與人員：林明言　國立交通大學資訊工程學系

一、中文摘要

循序樣式的勘測，是從序列資料庫的序列中，挖掘出所有具循序關係的項目集。通常使用者必須指定一最小支持。然而，使用者通常必須不斷地嘗試各不同的最小支持數值並觀察勘測結果，經由這種互動式循序樣式勘測，找到理想的結果。本計畫建構一個高效率的互動式循序樣式勘測方法。以建立知識庫的方式，來改善互動式循序樣式探勘的效率。減少使用者各次探勘所需的執行時間，並減少整個互動式勘測所需之時間。我們所設計的資料結構與方法，可以提供有效率的、符合多使用者需求的、互動式循序樣式勘測。

關鍵詞：資料探勘、互動式勘測、知識庫

Abstract

The discovery of sequential patterns has become a challenging task due to its complexity. Essentially, a user would specify a minimum support threshold with respect to the database to find out the desired patterns. The mining process is usually iterative since the user must try various thresholds to obtain the satisfactory result. In order to minimize the total execution time and the response time for each trial, we propose a knowledge base assisted algorithm for interactive sequence discovery, called KISP. KISP constructs a knowledge base accumulating the pattern information in individual mining, eliminates considerable amount of potential patterns to facilitate efficient support counting, and speeds up the whole process. In addition, we further optimize the algorithm by direct generations of the reduced candidate sets and concurrent counting of variable sized candidates. The conducted experiments show that KISP outperforms GSP by several orders of magnitudes for interactive sequence discovery

Keywords: data mining, Interactive discovery, knowledge base

二、緣由與目的

Mining sequential patterns, which finds out temporal associations among item-sets in the sequence database, is an important issue in data mining. A classic application of the problem is the market basket analysis whose database contains purchase records, where each record is an ordered sequence of *itemsets* (sets of items) bought by a customer. The objective is to discover the itemsets in future purchase after certain itemsets were bought. The mining technique can be applied to various domains such as discovering the relationships between the symptoms and certain diseases in medical applications. In comparison to the mining of association rules [3], sequential pattern mining is more complicated because not only the frequent itemsets but also the temporal relationships must be found.

The mining process is very difficult and time-consuming because patterns could be formed by any permutation of itemsets formed by any combination of possible items in the

database. In order to distinguish the interesting patterns, a user must supply a *minimum support* threshold (abbreviated *minsup*) for the mining. The result of mining finds out the set of patterns having *supports* greater than or equal to the *minsup*. The *support* of a pattern is the percentage of sequences (in the database) containing the pattern. The discovered patterns are called sequential patterns or frequent sequences. Most approaches focused on minimizing the search space of *potential sequential patterns* (called *candidates*), or on minimizing the required disk I/O due to the multiple database scanning. All these approaches discover the patterns by directly executing the mining algorithms once a *minsup* is specified.

However, the mining process is typically iterative and interactive since a user may specify a *minsup* value that results in too many or too few patterns. Usually, the user must try various *minsups* until the result is satisfactory. Nevertheless, most approaches are not designed to deal with repeated mining under such circumstance so that each *minsup* invokes a re-mining from scratch. Some approaches solved the interactive problem by pre-processing using an assumed least *minsup*. Nevertheless, the lengthy pre-processing has to be executed again if a user supplies a *minsup* below the assumed least value.

Therefore, we propose a simple approach, called *KISP*, to improve the efficiency of sequential pattern discovery with changing supports. *KISP* utilizes the information obtained from prior mining processes, and generates a knowledge base (abbreviated *KB*) for further queries about sequential patterns of various *minsups*. When the results cannot be directly derived from the knowledge base, *KISP* incorporates *KB* into a fast sequence discovery by eliminating the candidates existing in *KB* before support counting. Unlike those approaches assuming a least *minsup* for pre-processing before iterative mining, *KISP* accepts any *minsup* value and has no difficulty in mining huge databases even with a small main memory. The conducted experiments on well-known synthetic data show that KISP effectively improves the interactive mining performance.

三、文獻回顧

The problem of interactive association discovery was addressed in [1]. The method in [1] preprocesses the data in the transactional database, and stores frequent itemsets in an adjacency lattice. Online repeated queries about association rules are answered by graph theoretic searching on the lattice.

Similarly, a knowledge cache is used for interactive association discovery in [9]. The knowledge cache contains frequent itemsets and the non-frequent itemsets, if memory space is available, that have been discovered while processing other queries. The study [9] indicated that their *benefit replacement* algorithm is the best caching algorithm.

Although on-line association discovery [1, 5, 9, 10] is close to our problem, these approaches aim to interactively find frequent itemsets rather than frequent sequences, which is more complicated. One related work of interactive sequence mining extended the *SPADE* algorithm [16] into the *ISM* (Incremental Sequence Mining) algorithm for incremental and interactive sequence mining [11]. All queries are performed on a pre-processed in-memory data structure, the Increment Sequence Lattice (*ISL*). Therefore, A ‘small enough’ *minsup* must be pre-selected to apply *SPADE* for pre-processing and saving the results in *ISL*. Nevertheless, if a query involves a threshold smaller than the pre-selected *minsup*, another (more) lengthy mining process must be performed to generate a new *ISL* for the new query. Moreover, as described in [11], the *ISM* might encounter memory problem if the number of the potentially frequent patterns is too large.
Without any assumption on the *minsup* value and on the required memory, the proposed algorithm speeds up interactive sequence discovery by using the acquired information with optimizations like direct candidate-generation and concurrent counting.

四、結果與討論

Fig. 1 outlines the *KISP* algorithm. We further optimized *KISP* by Theorem 1, which is used to generate the new-candidates in pass k (denoted by X_k') directly. **Theorem 1.** $X_k' = (S_{k-1}[KB.base] \otimes N_{k-1}[minsup]) \cup (N_{k-1}[minsup]$

⊗ N_{k-1}[*minsup*]). That is, X_k' is the union of the two sets; one obtained from joining the frequent (*k-1*)-sequences in *KB* with the new frequent (*k-1*)-sequences, the other obtained from self-joining the new frequent (*k-1*)-sequences. The concurrent support counting technique further minimizes the number of database scanning required by counting variable sized candidates concurrently in pass *k*

The relative performance of *KISP* and *GSP* is described below. Take *minsup* = 0.75% for example, the execution time ratio of *GSP* to *KISP* is 2.1 times. The time saved by *KISP* resulted from the reduced number of candidates—*GSP* counted 2.3 times the number of candidates. *KISP* exhibits excellent mining capability for query intensive applications. As we increased the number of queries from 3 to 11, the average execution time (also the time required for posterior queries) decreased from 1763 seconds to 514 seconds.

In the experiments with concurrent optimization, the number of database scanning reduced by concurrent support counting is 6, and the reduced execution time is 94 seconds for the mining with *minsup*=0.5%. Most scans were combined in pass three so that the total number of passes and the total execution times were reduced. When users need to find the appropriate set of patterns by reducing the number of patterns found in a query, the next specified *minsup* would be greater than the counting base of *KB* (*KB.base*). In the next experiment, all *KB.base*s of the *KB*s were 0.5%, and 100 *minsup*s ranging from 0.5% to 2.5% were randomly selected. The mining results are all available in very short time with average execution time 4.3 seconds and maximum execution time 22 seconds. For most queries, the execution time of *KISP* is several orders of magnitude faster than *GSP*, which always re-mines from scratch.

In the scale-up experiments, the total number of customers was increased from 100K to 1000K, running the same series of *minsup* (2.5% down to 0.5%). Since *KISP* retrieves merely S_{k-1}[*KB.base*] (i.e. frequent (*k-1*)-sequences in *KB*) for generating candidate *k*-sequences, even without large memory, *KISP* may efficiently discover patterns in large databases with *KB*. The execution time of *KISP* increases linearly as the database size increases.

五、計畫成果自評

The problem of interactive sequence mining is extensively studied in the project. The result is a satisfactory accomplishment, the KISP algorithm. The comprehensive experiments also show that the proposed algorithm outperforms current state-of-the-art algorithm and can be used to improve the mining efficiency of interactive sequence mining. We summarize this project in a paper, which is accepted in the HICSS-36 conference. This also confirms the project is successful.

六、參考文獻

[1] C. C. Aggarwal and P. S. Yu, "Online Generation of Association Rules," *Proceedings of the 14th International Conference on Data Engineering*, Orlando, Florida, USA, Feb. 1998, pp. 402-411.

[2] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proceedings of the 11th International Conference on Data Engineering*, Taipei, Taiwan, 1995, pp. 3-14.

[3] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, Sep. 1994, pp. 487-499.

[4] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal and M.-C. Hsu, "FreeSpan: Frequent pattern-projected sequential pattern mining," *Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 355-359.

[5] C. Hidber, *Online Association Rule Mining*, Technical Report UCB/CSD-98-1004, U. C. at Berkeley, 1998.

[6] M. Y. Lin and S. Y. Lee, "Incremental Update on Sequential Patterns in Large Databases," *Proceedings of 10th IEEE International Conference on Tools with Artificial Intelligence*, 1998, pp. 24-31.

[7] H. Mannila, H. Toivonen and A. I. Verkamo, "Discovery of Frequent Episodes in Event Sequences," *Data Mining and Knowledge Discovery*, Vol. 1, Issue 3, 1997, pp. 259-289.

[8] A. M. Mueller, *Fast Sequential and* Parallel *Algorithm for Association Rule Mining: A Comparison*, Technical report CS-TR-3515, University of Maryland, 1995.

[9] B. Nag, P. M. Deshpande and D. J. DeWitt, "Using a Knowledge Cache for Interactive Discovery of Association Rules," *Proceedings of the 1999 SIGKDD Conference*, San Diego, California, Aug. 1999, pp.

244-253.
[10] S. Parthasarathy, S. Dwarkadas and M. Ogihara, "Active Mining in a Distributed Setting," *Proceedings of Workshop on Large-Scale Parallel KDD Systems*, San Diego, CA, USA, Aug. 1999, pp. 65-85.
[11] S. Parthasarathy, M. J. Zaki, M. Ogihara, and S. Dwarkadas, "Incremental and interactive sequence mining," *Proceedings of the 8th International Conference on Information and Knowledge Management*, Kansas, Missouri, USA, Nov. 1999, pp. 251-258.
[12] J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal and M.-C. Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-projected Pattern Growth," *Proceedings of 2001 International Conference on Data Engineering*, 2001, pp. 215-224.
[13] T. Shintani and M. Kitsuregawa, "Mining Algorithms for Sequential Patterns in Parallel: Hash Based Approach," *Proceedings of the Second Pacific–Asia Conference on Knowledge Discovery and Data mining*, 1998, pp. 283-294.
[14] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proceedings* of the *5th International Conference on Extending Database Technology*, Avignon, France, 1996, pp. 3-17.
[15] K. Wang, "Discovering Patterns from Large and Dynamic Sequential Data," *Journal of Intelligent Information Systems*, Vol. 9, No. 1, 1997, pp. 33-56.
[16] M. J. Zaki, "Efficient Enumeration of Frequent Sequences," *Proceedings of the 7th International Conference on Information and Knowledge Management*, Washington, USA, Nov.1998, pp. 68-75.

Algorithm KISP (*DB*, *KB*, *minsup*)
Input: *DB* = the database of data sequences; *minsup* = user specified minimum support ;
KB = knowledge base having the supports of all the candidates in prior minings
Output : *S*[*minsup*] = sequential patterns with respect to *minsup*; *KB* = (new) knowledge base

// Let *x.sup* be the support of a candidate x , X_k[*minsup*] be the set of candidate *k*-sequence in *DB* with
// respect to *minsup, and KB.base* be the counting base (the smallest *minsup* used) in constructing the *KB*
1) if $KB = \phi$ then $KB = \{x \text{ and } x.sup, \forall x \in X_1\}$;
2) $S[minsup] = \{x \mid x \in KB \wedge x.sup \geq minsup\}$; // obtain valid sequential patterns from knowledge base
3) if *minsup* < *KB.base* then // mine new patterns and accumulate new knowledge
4) $k = 2$;
5) generate $X_k[minsup]$ from the frequent (*k-1*)-sequences in $S[minsup]$;
6) $X_k' = X_k[minsup] - \{x \mid x \in KB\}$; // eliminate those candidate *k*-sequences in *KB*
7) while $X_k' \neq \phi$ do // there exist candidate *k*-sequences, obtains their supports
8) forall data sequences *ds* in database *DB* do
9) for each candidate $x \in X_k'$ do
10) increase the support of *x* if *x* is contained in *ds* ;
11) endfor
12) endfor
13) $KB = KB \cup \{x \text{ and } x.sup, \forall x \in X_k'\}$; // collect new candidates and their supports
14) $S[minsup] = S[minsup] \cup \{x \mid x.sup \geq minsup \wedge x \in X_k'\}$; // collect new patterns from X_k'
15) $k = k+1$;
16) generate $X_k[minsup]$ from the frequent (*k-1*)-sequences in $S[minsup]$;
17) $X_k' = X_k[minsup] - \{x \mid x \in KB\}$; // the reduced set eliminates candidate *k*-sequences in *KB*
18) endwhile
19) *KB.base* = *minsup* ; // update the counting base of *KB*
20)endif

Fig. 1. Algorithm *KISP*