Mining Interval Sequential Patterns

Ding-An Chiang,[1,†] Shao-Lun Lee,[2,*] Chun-Chi Chen,[1,‡] Ming-Hua Wang[3,§]
[1]*Department of Information Engineering, Tamkang University, Tan-Shui, Taipei, Taiwan, R.O.C.*
[2]*Department of Information Management, Oriental Institute of Technology, Pan-Chiao, Taipei, Taiwan, R.O.C.*
[3]*Department of Information Management, Nanya Institute of Technology, Jung-Li, Taoyuan, Taiwan, R.O.C.*

The main task of *mining sequential patterns* is to analyze the transaction database of a company in order to find out the priorities of items that most customers take when consuming. In this article, we propose a new method—the *ISP Algorithm*. With this method, we can find out not only the order of consumer items of each customer, but also offer the periodic interval of consumer items of each customer. Compared with other previous *periodic association rules*, the difference is that the period the algorithm provides is not the repeated purchases in a regular time, but the possible repurchases within a certain time frame. The algorithm utilizes the transaction time interval of individual customers and that of all the customers to find out when and who will buy goods, and what items of goods they will buy.

1. INTRODUCTION

In the 21st century, business owners deeply understand the meaning of Charles Darwin's famous saying "survival of the fittest." The competition among businesses became white hot. We have "upgraded" from a traditional information war to an all-round intelligence war. The strategy "providing the most information to users" in the past to help users to make decisions no longer copes with unpredictable fluctuations on the customer market. The ability to handle the target market more precisely and promote enterprises' competitive intelligence become an indispensable weapon for the enterprises.

Moreover, with the coming of the new knowledge era, everything needs efficiency and effect. So does a successful business. To make a decision, decision makers in the enterprise have to analyze many reports and much information (both

*Author to whom all correspondence should be addressed: e-mail: sllee@mail.oit.edu.tw.
[†]e-mail: chiang@cs.tku.edu.tw.
[‡]e-mail: mauroce.chen@msa.binet.net.
[§]e-mail: wangmh@nanya.edu.tw.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 20, 359–373 (2005)
 • DOI 10.1002/int.20070

in the enterprise and out of the market). However, they need equivalent knowledge and experiences to handle this, because information analysis is a complicated and difficult work. When collecting information, one may need to gather it from many possible sources. According to different origins, the management information is analyzed by different computer programs that have been designed by programmers. The problem is how to find out unobvious but valuable information from databases and provide it to decision makers for reference.

The sequential pattern can be divided into Sequential Procurement[1] and Cyclic Procurement[2] by the sequence and the section of time. What sequential procurement concerns is the chronological pattern. It only analyzes the association of data by the sequence of time. It focuses on searching items or merchandise that customers keep purchasing, and then suggests business to promote new products of the kind to the customers. Lawrence et al.[3] use Association Rule and Clustering to analyze past consuming data of customers, hoping to group customers properly. From analysis, business can understand consuming forms of each customer and then recommend suitable goods to customers who are interested to raise the response rate of commercial merchandise.[3] Kitts et al.[4] use the concept of cross sale to calculate the number of the customers who have bought certain goods and the possibility that they will buy some other merchandise afterward. According to the purchase probability, then one can decide what items of goods to recommended to the customers.[4] On the other hand, what cyclic procurement concerns is the events that happen in the same time section and whether they reoccur in other time sections as well. It emphasizes the variation in different time sections. Generally, when examining the time window of the related rules that are found in the transactional database, we may find that some rules appear at a certain time of a year, maybe in a few months or a couple of weeks, and show their periodicities. Ozden et al. proposed the Sequential Algorithm and Interleaved Algorithm to determine cyclic association rules.[2] All the methods introduced above are helpful to decision makers to find out valuable information and they have practical values in marketing management.

This article uses the concept mentioned above, combining data mining skills and fundamentals of statistics, to develop a new algorithm that we called Interval Sequential Pattern—the ISP Algorithm. It can find out not only the order of purchase items of each customer, but also the time interval of the periodic purchase items. By using the time interval information provided by this sequential pattern, marketing people can sell suitable products to suitable customers at the suitable sale time and reach enterprise competitive intelligence.[5,6]

The ISP Algorithm that we propose in this article is different than the two types mentioned above. The difference is that order patterns only show the priority order of purchased items. For example, Rule A $\rightarrow$ B $\rightarrow$ C [0.20] represents the purchase order of the customer who, after buying item A, will buy item B and then item C in a row. Twenty percent of the customers meet such a purchase order, but the rule cannot show the time interval of the sequential pattern, that is, the traditional sequential pattern cannot show how much time it takes from the sale of the first item A to that of the last item C; therefore, we cannot know when is a suitable time to recommend product C after the customer buys product A.

What is different with the cyclic association rule is that the sequential pattern that is found is not merely the periodic purchase of items; therefore, we named the algorithm the Interval Sequential Pattern instead to emphasize whether or not it is a periodic purchase. The algorithm uses the fundamentals of statistics to calculate the shortest time interval of purchase, the longest time interval of purchase, and the average time interval of purchase in each rule. It helps marketing people to recommend suitable merchandise to customers at the suitable time. Also, the ISP Algorithm separately calculates the shortest time interval, the longest time interval, and the average time interval in the transactional sequential pattern of each individual along with those of overall customers. It shows the different consuming behavior between individuals and the whole, and separates and groups different customers further. So we can use the rule we found to analyze the model of customers' consuming behavior and to help us to make a further profitable decision. According to this result, we can compare the different consuming behaviors between individual and overall customers, and provide more information to help marketing people to make decisions in advance. Enterprises that get such information can be the master of the likes of customers more precisely and reach target marketing.

2. RELATED RESEARCH

As pointed out by Agrawal and Srikant,[1] in the associated definition of the Mining Sequential Pattern, a sequence has multi-itemsets, and is arranged in some certain order. We represent a sequence as $\langle s_1, s_2, \ldots, s_n \rangle$, and s_i is an itemset in it. If two sequences $\langle a_1, a_2, \ldots, a_n \rangle$ and $\langle b_1, b_2, \ldots b_m \rangle$ have an order integer $i_1 < i_2 < \cdots < i_n$, $1 \leqq i_k \leqq m$, and make $a_1 \subseteq b_{i1}, \ldots, a_n \subseteq b_{in}$, then we say sequence $\langle b_1, b_2, \ldots, b_m \rangle$ includes sequence $\langle a_1, a_2, \ldots, a_n \rangle$, and at the same time, $\langle a_1, a_2, \ldots, a_n \rangle$ is $\langle b_1, b_2, \ldots, b_m \rangle$ and the length of it is a subsequence of n. In sequence set, if a sequence s is the maximal sequence, then sequence s is not included by other sequences. A customer sequence is the whole number of itemsets of a customer, the sequence that is sorted by the order of priority of transaction time. If sequence s is included by a customer sequence, then we say that the customer supports sequence s. A support for a sequence is the number of the overall customers that support the sequence. If the support of a sequence fits the minimal support, then this sequence is called a frequent sequence. Otherwise, it is called a nonfrequent sequence. The length of a sequence consists of the number of itemsets of that sequence. We denote k-sequence for the sequence of the length k, and a frequent sequence of length k is indicated as k-frequent sequence.

In general, there are two phases to create the sequential pattern: setting the large itemset and establishing the sequential pattern rules by large itemset.[7,8] As in the association rule,[9,10] the meaning of sequential patterns is similar to the large itemset, which is to find out the association between items. The only difference is that in sequential patterns, the chronological pattern is one of the major concerns. In phase I, the Apriori Algorithm is applied to find the large itemset. Whereas in phase II, there are five phases to generate the sequential pattern rule from the large itemset.

Table I. The customer sequence is ordered by increasing transaction time.

Transaction ID	Transaction date	Customer ID	Items
10	2001/10/02	ID0001	10
11	2001/11/06	ID0001	30
12	2001/11/15	ID0001	40,80
13	2001/11/20	ID0001	20,30
15	2001/12/02	ID0001	30,80
17	2001/12/04	ID0001	40
07	2001/07/22	ID0002	20
08	2001/08/06	ID0002	30
09	2001/08/16	ID0002	40,70,80
14	2001/11/25	ID0002	30
16	2001/12/20	ID0002	40,70
18	2001/12/25	ID0002	30,70
19	2001/12/28	ID0002	40
03	2001/03/26	ID0003	30,50
06	2001/04/20	ID0003	60
04	2001/03/27	ID0004	10,20
05	2001/04/02	ID0004	90
01	2001/02/03	ID0005	30
02	2001/02/25	ID0005	40,70

1. Sort Phase: Customer series is the major figure whereas transaction time is the sub-major figure to arrange the raw data from the original transaction database. The new database after sorting incrementally by time is shown in Table I.
2. Large Itemset Phase: Find out all the large itemsets and denote each large itemset with a specific symbol.
3. Transformation Phase: Delete the nonlarge itemsets from the raw transaction database. That is, the items not in the large itemset from the raw transaction database are deleted and the data rearranged by customers.
4. Sequential Phase: Delete the sequences that each contain two items from the data. The sequences can be generated by an algorithm similar to Apriori.
5. Maximal Phase: Find out the maximal sequences within the frequent sequences, to link up the sequence phase to reduce the time of counting nonmaximal sequences. The sequence generated in the previous step that is not contained in others is the largest sequence.

3. INTERVAL SEQUENTIAL PATTERNS

3.1. Calculating the Transaction Time Interval

In the original transactional database, we need to discuss the different transactional behavior model of each customer separately; therefore, according to the transaction sequence of each customer, we have to calculate separately the shortest time interval, the longest time interval, and the average time interval of each customer. By doing this, we can provide marketing people the most suitable time to sell that merchandise. The following are the associated definitions and the formula for calculation.

DEFINITION 1: TRANSACTION TIME INTERVAL. *Let a customer sequence be* $C = \langle C_1, C_2, \ldots, C_i, \ldots, C_m \rangle$ *and the transaction dates be* $D = \langle D_1, D_2, \ldots, D_i, \ldots, D_m \rangle$, $1 \leqq i \leqq m$. *If there exists a sequential pattern* $SP = \langle S_1, S_2, \ldots, S_n \rangle$, *such that* $S_1 = C_{j+1}, S_2 = C_{j+2}, \ldots, S_n = C_{j+n}$, *where* $0 \leqq j \leqq (m - n)$ *and* $1 \leqq n \leqq < m$, *then the transaction time interval* $TI = D_{j+n} - D_{j+1}$. *Otherwise, if the sequential pattern does not exist, then the transaction time interval* $TI = 0$.

Take Table I as an example: The whole transaction time interval of sequential pattern ⟨(30)(40)⟩ in the customer sequence ⟨(10)(30)(40,80)(20,30)(30,80)(40)⟩ of Customer No. ID0001 is 9 days (2001/11/15–2001/11/06) and 34 days (2001/12/24–2001/11/20); with the same method, the whole transaction time interval of sequential pattern ⟨(30)(40)⟩ in the customer sequence ⟨(20)(30)(40,70,80)(30)(40,70)(30,70)(40)⟩ of Customer No. ID0002 is 10 days (2001/08/16–2001/08/06), 25 days (2001/12/20–2001/11/25), and 3 days (2001/12/28–2001/12/25); however, with Customer No. ID0003, the sequential pattern ⟨(30)(40)⟩ is not included in the sequential pattern ⟨(30,50)(60)⟩, so the transaction time interval is 0 days. By using the same method, with Customer No. ID0004, because the sequential pattern ⟨(30)(40)⟩ is not included in the sequential pattern ⟨(10,20)(90)⟩, the transaction time interval is 0 days; the sequential pattern ⟨(30)(40)⟩ in the customer sequence ⟨(30)(40,70)⟩ of Customer No. ID0005 is 22 days (2001/02/25–2001/02/03).

In addition, from Definition 1 we can calculate the transaction time interval of individual customers; the following are the associated definitions.

DEFINITION 2: TRANSACTION TIME INTERVAL OF INDIVIDUAL CUSTOMERS:

- *The shortest transaction time interval of individual customers: If sequential pattern* $SP = \langle S_1, S_2, \ldots, S_n \rangle$ *appears in customer sequence* $C = \langle C_1, C_2, \ldots, C_m \rangle$ *one or more than one time, then the shortest transaction time interval of the individual customer is the minimum of the transaction time interval (if it contains nonzero values, the minimum of the transaction time interval must be greater than zero) that appeared in the sequential pattern.*
- *The longest transaction time interval of individual customers: If sequential pattern* $SP = \langle S_1, S_2, \ldots, S_n \rangle$ *appears in customer sequence* $C = \langle C_1, C_2, \ldots, C_m \rangle$ *one or more than one time, then the longest transaction time interval of the individual customer is the maximum of the transaction time interval that appeared in the sequential pattern.*
- *The average transaction time interval of individual customers: If sequential pattern* $SP = \langle S_1, S_2, \ldots, S_n \rangle$ *appears in customer sequence* $C = \langle C_1, C_2, \ldots, C_m \rangle$ *one or more than one time, then the average transaction time interval of the individual customer is the average of the transaction time interval that appeared in the sequential pattern.*

Take Table I as an example, the overall transaction time interval of sequential pattern ⟨(30)(40)⟩ in the customer sequential pattern of Customer No. ID0001 is 9 days and 34 days; therefore, the shortest time interval = min(9,34) = 9 days, the longest time interval = max(9,34) = 34 days, and the average time interval = (9 + 34)/2 = 21.5 days. The overall transaction time interval in the customer sequential pattern of Customer No. ID0002 is 10 days, 25 days, and 3 days; therefore, the shortest time interval = min(10,25,3) = 3 days, the longest time

interval = max(10,25,3) = 25 days, and the average time interval = (10 + 25 + 3)/3 = 12.67 days. The overall transaction time interval in the customer sequential pattern of Customer No. ID0003 is 0 days; therefore, the shortest time interval = min(0) = 0 days, the longest time interval = max(0) = 0 days, and the average time interval = 0 days. The overall transaction time interval in the customer sequential pattern of Customer No. ID0004 is 0 days; therefore, the shortest time interval = min(0) = 0 days, the longest time interval = max(0) = 0 days, and the average time interval = 0 days. The overall transaction time interval in the customer sequential pattern of Customer No. ID0005 is 22 days; therefore, the shortest time interval = min(22) = 22 days, the longest time interval = max(22) = 22 days, and the average time interval = 22/1 = 22 days.

The overall customer transactions are different from individual customer transactions. It discusses the transaction behavior models of the whole customers. These overall customer transaction models are generated from the transaction behavior models of individual customers to form a common rule, which has the same transaction behavior models as those of overall customers. In other words, the rule that is generated by the overall customers' transaction sequential pattern must be larger than one of the minimal support; namely, the rule will only appear while the amount of individual customers have the transaction behavior models. The following definitions are associated with "Transaction time interval of overall customers."

Definition 3: Transaction Time Interval of Overall Customers:

- *The shortest transaction time interval of overall customers: The shortest transaction time interval of overall customers of sequential pattern* $SP = \langle S_1, S_2, \ldots, S_n \rangle$ *is the minimum of the shortest transaction time interval of all individual customers (if it contains nonzero values, the minimum of the shortest transaction time interval must be greater than zero).*
- *The longest transaction time interval of overall customers: The longest transaction time interval of overall customers of sequential pattern* $SP = \langle S_1, S_2, \ldots, S_n \rangle$ *is the maximum of the longest transaction time interval of all individual customers.*
- *The average transaction time interval of overall customers: The average transaction time interval of overall customers of sequential pattern* $SP = \langle S_1, S_2, \ldots, S_n \rangle$ *is the weighted mean of the average transaction time interval of all individual customers.*

As in Table I, the shortest transaction time interval of overall customers in sequential pattern $\langle (30)(40) \rangle$ is equal to min(9,3,0,0,22) = 3 days, the longest transaction time interval of overall customers is equal to max(34,25,0,0,22) = 34 days, and the average transaction time interval of overall customers is equal to $(21.5 \cdot 2 + 12.67 \cdot 3 + 0 \cdot 0 + 0 \cdot 0 + 22 \cdot 1)/(2 + 3 + 0 + 0 + 1) = 17.17$ days.

3.2. ISP Algorithm

As shown in Section 2, the complicated work in the problems of mining sequential patterns is at the fourth step, the sequence phase, that is, find out all the frequent sequences. Therefore, our method emphasizes how to calculate the shortest time interval, the longest time interval, the average time interval, number

of appearances, and the support of each sequential pattern in each customer sequence in the phase of sequence. As for other work phases, we follow the steps in Section 2 without changing the steps very much.

The principle of the process of the algorithm we proposed is to repeat the calculation constantly for many times. Each round includes three steps and the final goal of each round is to calculate the frequent sequence, which is one length longer than the last result, and it ends when the ISP cannot produce any longer length of the frequent sequence. The corresponding algorithm is shown in Figure 1.

The processing of each step is explained in detail as follows: Let FS_k be the set of k-frequent sequence, FS be the total set of the all frequent sequences, and CS_k be the set of k-candidate sequences.

- Step 1: Producing Candidate sequence CS_k ($k = 2,3,4\ldots$). The first step of each round must generate the candidate sequence of this round based on the $(k-1)$-frequent sequence set of the last round.

  The first time, the length 2 that is the candidate sequence set of CS_k ($k = 2$) is generated as follows: For any two items s_1 and s_2 both belonging to FS_1, and $s_1 \neq s_2$, it generates an outcome that $\langle s_1, s_2\rangle$ and $\langle s_2, s_1\rangle$ belong to CS_2.

  The way that CS_k ($k > 2$) is generated in other times is as follows: For any two $(k-1)$-frequent sequences $x_1 = \langle a_1,\ldots,a_{k-2},a_{k-1}\rangle$ and $y_1 = \langle b_1,\ldots,b_{k-2},b_{k-1}\rangle$. If $a_1 = b_1,\ldots,a_{k-2} = b_{k-2}$, we can generate two k-frequent sequences $\langle a_1,\ldots,a_{k-2}, a_{k-1},b_{k-1}\rangle$ and $\langle a_1,\ldots,a_{k-2},b_{k-1},a_{k-1}\rangle$.

  The action following Step 1 is to check whether CS_k is an empty set. If CS_k is an empty set, ISP stops; this is because there is no more frequent sequence being generated in this time. It means frequent sequences of all lengths are all found. Otherwise, we delete those that cannot become candidate sequences of k-frequent sequence. In other words, if a k-sequence wants to become a frequent sequence, all the subsequences of the length $k-1$ must be frequent sequences.

  As for every s in each candidate sequence CS_k, each s subsequence has to be checked individually to see if it appears in FS_{k-1}. If not, we shall delete s from CS_k.

  After deleting, if CS_k becomes an empty set, ISP stops, or we will go on to the next step.
- Step 2: Generating the shortest, the longest, and the average transactional time interval of individual customers and its frequency.
  Notation:
  TTIIC: abbreviation of "transactional time interval of individual customers"
  min_t: the shortest *TTIIC*
  max_t: the longest *TTIIC*
  count: the frequency of transaction of individual customers
  total: the sum of *TTIIC*
  avg_t: the average of *TTIIC*.

  The first step is to calculate the number n, where n represents the number of the first item s_1 of each k-candidate sequence $s = \langle s_1, s_2,\ldots,s_k\rangle$ shown on every customer sequence $c = \langle c_1, c_2,\ldots,c_m\rangle$. Thus by using a loop to search n, the position and transaction time $t(s_1)$ of item s_1 in customer sequence can be found.

  In addition, to find the next item forwardly use the position and the time to check whether the k-candidate sequence $s_2,\ldots,s_k$ appears in customer sequence c or not. If it does, record the position and transaction time $t(s_k)$ of item s_k in the customer sequence and then calculate the transaction time interval $t(s_k) - t(s_1)$ of individual customers from s_1 to s_k.

  To compare the *TTIIC* with the shortest *TTIIC min_t*, if $t(s_k) - t(s_1) < min\_t$, then $min\_t = t(s_k) - t(s_1)$.

```
FS1={Frequent 1-sequences};
For (k=2; FSk-1≠∅; k++) do begin
      CSk = Apriori-All (FSk-1);
      ISP_individual_customers(CSk)
      ISP_overall_customers(CSk)
End

Apriori-All()
{
Insert into CSk
Select p.item1, p.item2, … , p.itemk-1, q.itemk-1
From FSk-1 p, FSk-1 q
Where p.item1 = q.item1, … , p.itemk-2 = q.itemk-2, p.itemk-1 < q.itemk-1;
Insert into CSk
Select p.item1, p.item2, … , q.itemk-1, p.itemk-1
From FSk-1 p, FSk-1 q
Where p.item1 = q.item1, … , p.itemk-2 = q.itemk-2, p.itemk-1 < q.itemk-1;
For itemsets i ∈ CSk do
      For all (k-1)-subsets s of i do
            If (s ∉ FSk-1) then
               Delete i from CSk
}

ISP_individual_customers()

{

min_t = 32767, max_t = 0, count = 0, total = 0, avg_t = 0

While there are k-candidate sequence s  <s1,s2,…,sk> generated do

begin

   While there are every customer sequence c=<c1,c2,…,cm> do

   begin

      Select Count(c.*) As n
      From c,s
      Where c.item = s1;

      Foreach n do

      begin

         Select c.timej+1 As t(s1), c.timej+2 As t(s2),…, c.timej+k As t(sk)
         From c,s
         Where c.itemj+1 = s1, c.itemj+2 = s2, … , c.itemj+k = sk;

         If t(sk)- t(s1) < min_t then min_t = t(sk) - t(s1)

         If t(sk)- t(s1) > max_t then max_t = t(sk) - t(s1)

         total = total + t(sk) - t(s1)

      End
```

Figure 1. ISP Algorithm. (*Figure continues on the following page.*)

```
    If total > 0 then
      count = count + 1
    Else
      Exit
    End
  End
  If count > 0 then avg_t = total / count
End
}

ISP_overall_customers()
{
t_min_t = 32767, t_max_t = 0, t_count = 0, t_total = 0, t_avg_t = 0
While there are k-candidate sequence s=<s1,s2,...,sk> generated do
begin
  While there are every customer sequence c=<c1,c2,...,cm> do
  begin
    Select c.shortesttime As min_t, c.longesttime As max_t, c.averagetime As ave_t, c.count As count
    From c,s
    Where c.item_{j+1} = s1, c.item_{j+2} = s2, ... , c.item_{j+k} = sk;
    If min_t < t_min_t then t_min_t = min_t
    If max_t > t_max_t then t_max_t = max_t
    t_total = t_total + ave_t * count
    temp = temp + count
    If count >0 then t_count = t_count + 1
  End
  If temp > 0 then t_avg_t = t_total / temp
  If t_count >= Min_support then FS_k = FS_k ∪ s
End
}
```

Figure 1. *Continued.*

To compare the *TTIIC* with the longest *TTIIC max_t*, if $t(s_k) - t(s_1) > max\_t$, then $max\_t = t(s_k) - t(s_1)$.

The number of transactions of individual customers $count = count + 1$, and the sum of *TTIIC total* $= total + t(s_k) - t(s_1)$.

After finding n times in this loop, we calculate average *TTIIC* $avg\_t = total/count$.

- Step 3: Generating the shortest time interval, the longest time interval, the average time interval, the support, and frequent sequence FS_k ($k = 2,3,4,\ldots$) of the transaction of the whole customers.

Notation:
TTIWC: abbreviation of "transactional time interval of whole customers"
t_min_t: the shortest *TTIWC*
t_max_t: the longest *TTIWC*
t_count: the number of individual customer present in the *k*-candidate sequence
t_total: the sum of *TTIWC*
t_avg_t: the average of *TTIWC*

First, select the shortest *TTIIC min_t*, the longest *TTIIC max_t*, transaction counts, and average *TTIIC ave_t* of every *k*-candidate sequence s_i (i represent the i *k*-candidate sequence) in every customer sequence c_j (j represent the j customer sequence) form the transaction database.

To compare the shortest *TTIIC min_t* with the shortest *TTIWC t_min_t*, if $min\_t < t\_min\_t$, then $t\_min\_t = min\_t$.

To compare the longest *TTIIC max_t* with the longest *TTIWC t_max_t*, if $max\_t > t\_max\_t$, then $t\_max\_t = max\_t$.

After that, calculate the average of *TTIWC t_avg_t* which equals the weighted mean of the average of *TTIIC*.

After comparing every *k*-candidate sequence s_i with every individual customer sequence c_j, the shortest *TTIWC t_min_t*, the longest *TTIWC t_max_t*, and average *TTIWC t_avg_t* can be generated.

One can tell how many customer sequences there are in which the count is larger than 1 by the number of counts that appear in each individual customer sequence c_j.

By calculating the support of the *k*-candidate sequence, FS_k can finally be generated by comparing the minimum support.

As mentioned before, the complicated task of the ISP Algorithm is in the fourth step, the sequence phase. Therefore, the efficiency of the sequence phase affects the executing time of examining the whole interval sequential pattern. In calculating the complexity of the time interval, supposing the biggest large itemset we found includes k items, we need to scan the database $(k + 1)$ times, and we have to calculate the time interval of every customer sequence every time we scan the database. The time we spent here at this part is $O(CS_k) \times O(n)$, where CS_k is the number of every *k*-candidate sequence, n is the number of times that the first item of the candidate sequence appeared in a customer sequence. Therefore, the maximum of the whole time complexity is $O(k \times t \times r \times n)$, where k is the length of the biggest large itemset, t is the number of transactions in the database, r is the total of rules being generated, n is the number of a item that only shows on a customer sequence. The associated efficiency analysis will be discussed in detail in the next section.

4. THE EXPERIMENT RESULTS

To examine the executing efficiency of the Interval Sequential Pattern Algorithm, we separately designed some experiments to examine our method, and

compared it with the AprioriAll Algorithm. We will introduce experimental environments, including the experiment desktop, the practicing program language, and information resources. In Section 4.1, we will introduce the experiment results. Then, in Section 4.2, we will compare the executing efficiency of the Interval Sequential Pattern Algorithm.

The following are the associated experimental environments:

1. Experimental platform
   - CPU: Pentium IV 2.4 GMHz
   - Memory: 1 GB RAM, 80 GB HD
   - Operating system: Windows 2000
   - Database: SQL Server 2000
2. Implementation language
   - Visual Basic 6.0
3. Experimental data source
   - We analyze the customers' data from a famous retail business in Taiwan, totaling more than 1.5 million transaction data, and more than 18 thousand customer's data from 2000 to 2001.

4.1. Interval Sequential Pattern Algorithm versus AprioriAll Algorithm

The executing frame of the AprioriAll Algorithm is shown in Figure 2; this method only provides the purchase priority of the all customers and the number of people who have done such a transaction behavior (support), but on the executing

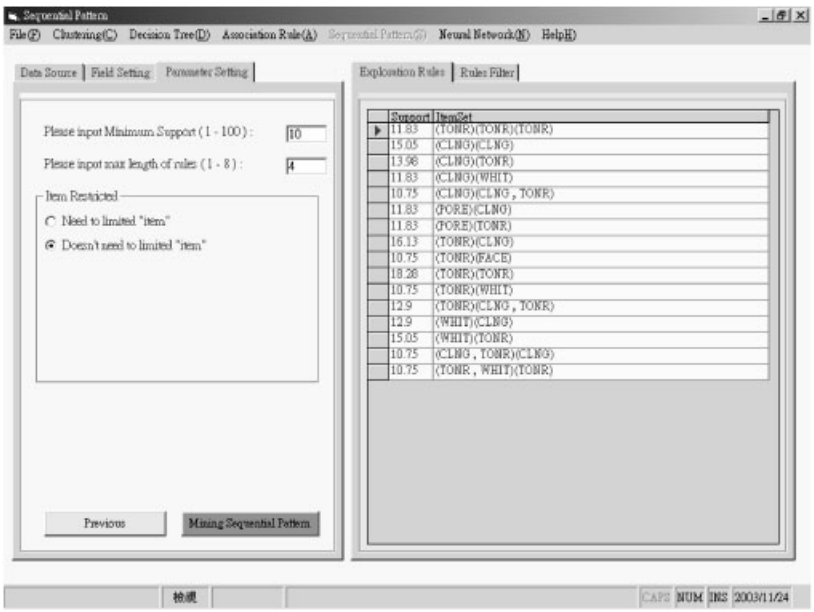

Figure 2. The executing frame of the AprioriAll Algorithm.

frame of the ISP Algorithm as shown in Figure 3, the algorithm we propose obviously can provide more information. It knows not only the support of every sequential pattern, but also the shortest, the longest, and the average time interval of the sequence in the whole customer transaction sequence. Also, if we want to know what kinds of customers have the same purchase order, we can see further detailed information as in Figure 4. In this figure, we can get the shortest time interval, the longest time interval, the average time interval, and the number of transactions of the sequential pattern in the transaction sequence of the individual customers. According to the date, marketing people know at what opportunity they can promote intended merchandise to the intended customers.

Take Figure 3 as an example: The marked place represents the (TONR) (TONR) (CLNG) rule, which means that after buying (TONR), the customers in that group will buy (TONR) again, and then buy (CLNG). Some 10.75% of the customers have such a purchase order and among these customers, some bought (TONR) again and (CLNG) within 49 days, and it will not be longer than 596 days. The average purchase time interval of the general customers who fit this purchase order is 263 days. If we want to inquire how many customers have this (TONR) (TONR) (CLNG) priority order in advance, we can click the rule and browse the detailed information. As shown in Figure 4, there are 10 customers, 00000300, 00001300, 00002100, 00003100, 00010500, 00012700, 00020500, 00020600, 00020900, and 00021200, who fit the (TONR) (TONR) (CLNG) priority order. Therefore, we use the last purchase dates, the last time that each of the 10 customers bought (CLNG), to calculate how long has passed, and then comparing the

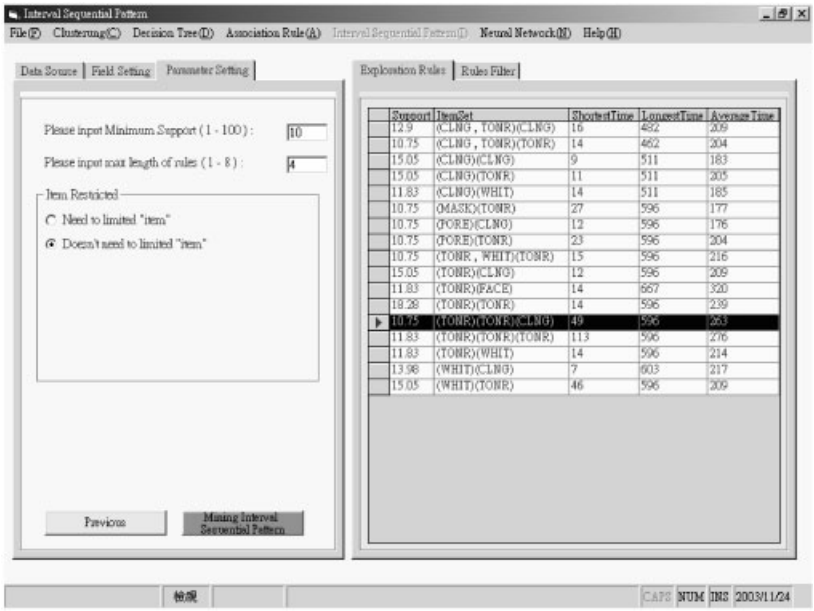

Figure 3. The executing frame of the ISP Algorithm.

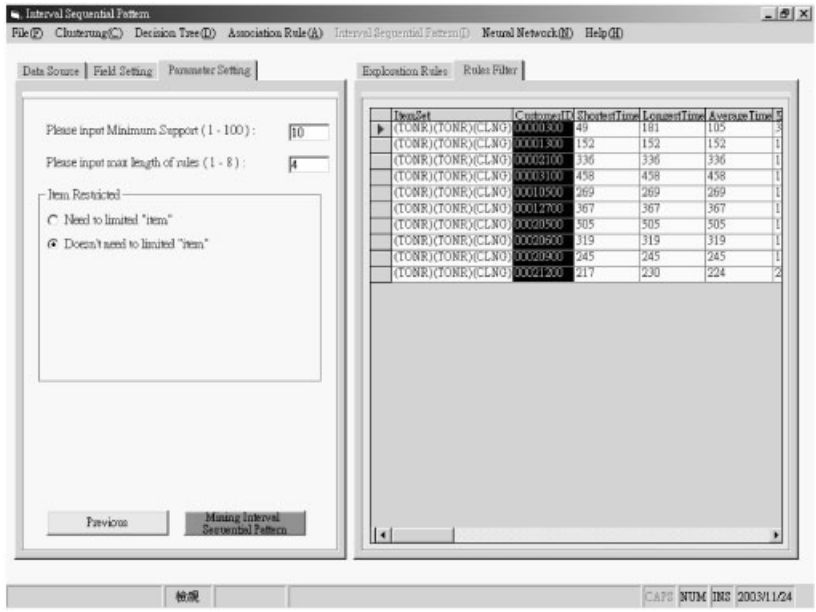

Figure 4. The executing frame of individual customers' detail information.

shortest, the longest, and the average transaction time interval of each customer, we finally come out with the recommended product purchase list of (TONR) and (CLNG) and so on.

4.2. Efficiency Analysis of the Interval Sequential Pattern Algorithm

To compare the executing efficiency of these two algorithms, we separately utilize four different supports, 0.05, 0.10, 0.15, and 0.20, and examine them under the condition of allowing that all the maximum lengths of rule are 4. The result are shown in Figure 5. The ISP Algorithm gives more information but has only slight differences when comparing the executing efficiency with AprioriAll Algorithm. It is because although the ISP Algorithm takes one more circuit time to calculate the time interval of every customer sequence (ISP Algorithm, Step 2), it generally only takes $O(n)$ more than the AprioriAll Algorithm. In the sequence, n is the number of the first item of the sequential pattern that appeared in the customer sequence.

5. CONCLUSION

In this article, we proposed a new algorithm called the ISP Algorithm. The greatest difference between the ISP Algorithm and the AprioriAll Algorithm is that the ISP Algorithm not only finds out the order of items that customers purchased and the number of people who have the same transactional behavior, but

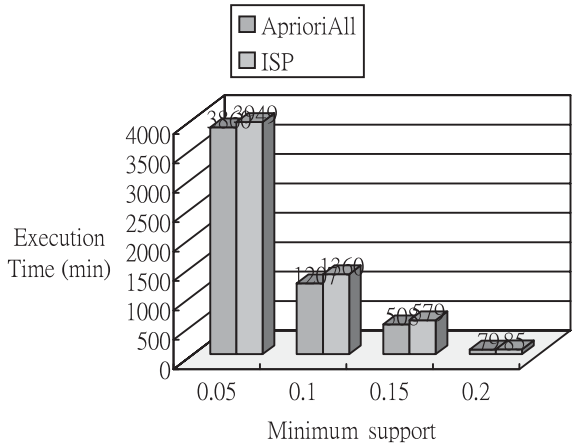

Figure 5. Execution time (same data set and different minimum support).

also provides the time interval customers spent from the first item to the last when purchasing. According to the result, we can compare customers' transactional records and predict the items a customer is going to buy (What: we can tell by the items in the sequential pattern), the time he or she will purchase (When: We can tell by the overall transactional intervals of customers), and the person who will follow this purchase order (Who: We can tell by the transactional details of each individual). With this method, the marketing people then can decide (What) commodities should be promoted to which customers (Who) at which timing (When), and it will gradually increase the response rate of the commercial merchandise and reach the marketing goal. Moreover, as to the overall efficiency, through this algorithm, one can obtain more information without spending much more extra time.

References

1. Agrawal R, Srikant R. Mining sequential patterns. In: Proc. 11th Int Conf Data Engineering, Taipei, Taiwan, March 6–10, 1995. pp 3–14.
2. Ozden B, Ramaswamy S, Silberschatz A. Cyclic association rules. In: Proc 14th Int Conf on Data Engineering, Orlando, Florida, February 23–27, 1998. pp 412–421.
3. Lawrence RD, Almasi GS, Kotlyar V, Viveros MS, Duri SS. Personalization of supermarket product recommendations. Data Min Knowl Discov 2001;5(1/2):11–32.
4. Kitts B, Freed D, Vrieze M. Cross-sell: A fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities. In: Proc ACM 6th Int Conf on Knowledge Discovery and Data Mining, Taipei, Taiwan, May 6–8, 2002. pp 437–446.
5. Fayyad UM, Shapiro GP, Smyth P. The KDD Process for extracting useful knowledge from volumes of data. Commun ACM 1996;39(11):27–34.
6. Tyson KWM. Competitive knowledge development: Reengineering competitive intelligence for maximun success. Compet Intell Rev 1995;6(4):14–21.

7. Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. In: Proc ACM SIGMOD Conf on Management of Data, Dallas, Texas, May 14–19, 2000. pp 241–250.
8. Tseng FC, Hsu CC. Generating frequent patterns with the frequent pattern list. In: Proc Asia Pacific Conf Data Mining and Knowledge Discovery, Hong Kong, April 16–18, 2001. pp 376–386.
9. Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proc 20th Int Conf on Very Large Data Bases, Santiago, Chile, September 12–15, 1994. pp 478–499.
10. Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large database. In: Proc ACM SIGMOD Int Conf on Management of Data, Washington, D.C., May 26–28, 1993. pp 207–216.