

Mining Closed Sequential Patterns with Time Constraints*

MING-YEN LIN, SUE-CHEN HSUEH⁺ AND CHIA-WEN CHANG

*Department of Information Engineering and Computer Science
Feng Chia University
Taichung, 407 Taiwan*

⁺*Department of Information Management
Chaoyang University of Technology
Taichung, 413 Taiwan*

The mining of closed sequential patterns has attracted researchers for its capability of using compact results to preserving the same expressive power as traditional mining. Many studies have shown that constraints are essential for applications of sequential patterns. However, time constraints have not been incorporated into closed sequence mining yet. Therefore, we propose an algorithm called CTSP for closed sequential pattern mining with time constraints. CTSP loads the database into memory and constructs time-indexes to facilitate both pattern mining and closure checking, within the pattern-growth framework. The index sets are utilized to efficiently mine the patterns without generating any candidate or sub-database. The bidirectional closure checking strategy further speeds up the mining. The comprehensive experiments with both synthetic and real datasets show that CTSP efficiently mines closed sequential patterns satisfying the time constraints, and has good linear scalability with respect to the database size.

Keywords: sequential pattern, closed sequential pattern, time constraint, memory index, sequence mining

1. INTRODUCTION

Sequential pattern mining [1, 2, 5, 14], which discovers frequent sub-sequences in a sequence database, is becoming an active research topic for its wide applications including customer behavior analysis [8, 9], web usage mining [6], DNA sequence analysis, *etc.* The discovered sequential patterns disclose not only the frequent items or events occurred together, but also the sequences of frequently appeared item-sets or events.

Many studies have shown that specifying constraints may increase the accuracy of mining results in practice. Various constraints, such as item, length, super-pattern, duration, and gap, can be specified to find more desirable patterns. Some constraints can be handled by a post-processing on the result of mining without constraints. However, time constraints affect the support computation of patterns so that they cannot be handled without adapting time attributes into the mining algorithms.

The issue of mining sequential patterns with time constraints was first addressed in [15]. Three time constraints including minimum gap, maximum gap and sliding time-window are specified to enhance the semantics of sequence discovery. For example, in a retail application, a discovered sequential pattern may be $\langle (TV)(Audio, Jukebox) \rangle$ which

Received February 2, 2007; accepted July 13, 2007.

Communicated by K. Robert Lai, Yu-Chee Tseng and Shu-Yuan Chen.

* This paper was supported by the National Science Council of Taiwan, R.O.C. under research grant No. NSC 95-2221-E-035-114.

means most customers could buy Audio and Jukebox after purchasing TV. However, if the time gap between two adjacent transactions in a pattern is long, such as over one year, the pattern could be useless for sales promotion. Specifying maximum gap helps to filter out such patterns. Moreover, if three days is indispensable for the supply of the stock and the time gap between two adjacent transactions in a pattern is less than two days, the pattern is unhelpful to replenish stocks in time. Minimum gap can be used so that the unhelpful patterns would not be displayed. Additionally, a consumer may forget to buy some goods in a transaction and buy them in the subsequent transaction within two days. A sliding window of two days may allow the two transactions to be handled (combined) as one. For instance, a data sequence $\langle_1(\text{TV})_7(\text{Audio})_9(\text{Jukebox})\rangle$ can support pattern $\langle(\text{TV})(\text{Audio}, \text{Jukebox})\rangle$. Therefore, adding time constraints for better mining results is an important problem.

In addition to time-constrained sequential pattern mining [8-10, 15, 19], the issue of mining sequential patterns has been extended. Various topics such as maximal sequential pattern mining, closed sequential pattern mining [4, 16, 17], top-k sequential pattern mining [13] and so on are studied. A sequential pattern is maximal if it is not a subsequence of another frequent sequence. It is closed if no any frequent super-sequence has the same support. For example, if sequential patterns with supports including $\langle(a)\rangle:3$, $\langle(c)\rangle:3$, $\langle(e)\rangle:2$, $\langle(a)(c)\rangle:2$, $\langle(a)(e)\rangle:2$, $\langle(c, e)\rangle:2$, $\langle(a)(c, e)\rangle:2$ are found, the closed patterns are $\langle(a)\rangle:3$, $\langle(c)\rangle:3$, $\langle(a)(c, e)\rangle:2$, and the maximal pattern is $\langle(a)(c, e)\rangle:2$ only. Although the closed patterns are more compact than the complete set of patterns, the same amount of information can be derived. For instance, the support of $\langle(a)(c)\rangle$, being 2, can be derived from $\langle(a)(c, e)\rangle$.

Many algorithms [2, 3, 7, 18] have been proposed to deal with the problem of mining sequential patterns. In general, these algorithms can be categorized into Apriori-like framework such as GSP [15] and SPADE [18], and pattern-growth framework such as PrefixSpan [11]. Although algorithms such as prefix-growth [12], DELISP [8], and cSPADE [19] may handle time constraints, they generally provide the complete set of time-constrained patterns, rather than the compact closed patterns. Algorithms CloSpan [17] and BIDE [16] successfully mine closed sequential patterns. However, the time constraint is yet to be incorporated. To the best of our knowledge, no algorithm has been presented to solve the problem of mining closed sequential patterns with time constraints. It is desirable to using compact results of such a mining to preserving the same expressive power as traditional mining.

Therefore, we propose an algorithm called CTSP (Closed Time-constrained Sequential Pattern mining) for mining closed sequential patterns with minimum gap, maximum gap, and sliding window constraints. Moreover, we define the closure of time-constrained patterns by contiguous sub/supersequences so that true time-constrained patterns with correct supports can be derived. CTSP utilizes the memory for efficient mining. Time-indexes are constructed in CTSP to facilitate both pattern mining and closure checking, within the pattern-growth framework. The closure checking is performed bidirectionally to speed up the mining. The extensive and comprehensive experiments with both synthetic and real datasets show that CTSP efficiently mines closed sequential patterns satisfying the time constraints, and has good linear scalability with respect to the database size.

2. PROBLEM STATEMENT

Let $\Psi = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ be a set of literals, called items. An itemset $I = (\beta_1, \beta_2, \dots, \beta_q)$ is a nonempty set of q items such that $I \subseteq \Psi$. A sequence s , denoted by $\langle e_1 e_2 \dots e_w \rangle$, is an ordered list of w elements where each element e_i is an itemset. Without loss of generality, we assume the items in an element are in lexicographic order. The length of a sequence s , written as $|s|$, is the total number of items in all the elements in s . Sequence s is a k -sequence if $|s| = k$. The sequence database DB contains $|DB|$ data sequences. A data sequence ds has a unique identifier sid and is represented by $\langle_{t_1} e_1' \ _{t_2} e_2' \ \dots \ _{t_m} e_m' \rangle$, where element e_i' occurred at time t_i , $t_1 < t_2 < \dots < t_m$.

A sequence s in the sequence database DB is a time-constrained sequential pattern (abbreviated as time-pattern) if $s.sup \geq minsup$, where $s.sup$ is the support of the sequence s and $minsup$ is the user specified minimum support threshold. The support of sequence s is the number of data sequences containing s divided by $|DB|$. Note that the support calculation has to satisfy three time-constraints $maxgap$ (maximum gap), $mingap$ (minimum gap), and $swin$ (sliding window). A data sequence $ds = \langle_{t_1} e_1' \ _{t_2} e_2' \ \dots \ _{t_m} e_m' \rangle$ contains a sequence $s = \langle e_1 e_2 \dots e_w \rangle$ if there exist integers $l_1, u_1, l_2, u_2, \dots, l_w, u_w$ and $1 \leq l_1 \leq u_1 < l_2 \leq u_2 < \dots < l_w \leq u_w \leq n$ such that the four conditions hold: (1) $e_i \subseteq (e_{l_i}' \cup \dots \cup e_{u_i}')$, $1 \leq i \leq w$; (2) $t_{u_i} - t_{l_i} \leq swin$, $1 \leq i \leq w$; (3) $t_{u_i} - t_{l_{i-1}} \leq maxgap$, $2 \leq i \leq w$; (4) $t_{l_i} - t_{u_{i-1}} \geq mingap$, $2 \leq i \leq w$. Assume that t_j , $mingap$, $maxgap$, $swin$ are all positive integers, $maxgap \geq mingap \geq 1$. Data sequence ds is a supersequence of sequence s (s is a subsequence of ds) if ds contains s . When $mingap$ is the same as $maxgap$, the time constraint is additionally called exact gap. Common sequential pattern mining without time constraints is a special case by setting $mingap = 1$, $maxgap = \infty$, $swin = 0$.

A sequence s is closed if no contiguous supersequence s_{sup} with the same support exists. Given a sequence $s = \langle e_1 e_2 \dots e_w \rangle$ and a subsequence s_{sub} of s , s_{sub} is a contiguous subsequence of s if s_{sub} can be obtained by any one of following ways: (1) dropping an item from either e_1 or e_w ; (2) dropping an item from element e_i ($1 \leq i \leq w$) which has at least two items; (3) s_{sub} is a contiguous subsequence of s_{sub}' , and s_{sub}' is a contiguous subsequence of s . Additionally, s is called a contiguous supersequence of s_{sub} . A sequence s is a closed time-constrained sequential pattern, abbreviated as closed time-pattern, if it is a time-pattern and is closed. The mining aims to discover the set of all closed time-patterns.

For example, given a sequence database DB of four data sequences with their $sids$ in Table 1, the rightmost column shows the closed time-patterns for constraints $mingap = 3$, $maxgap = 15$, $swin = 2$, and $minsup = 50\%$. In Table 1, data sequence C4 $\langle_{5(a)} (d)_{10} (c)_{21} \rangle$ has three itemsets occurring at time 5, 10, 21, respectively. The third itemset of C4 has two items c and d . Sequences $\langle (a, c)(b) \rangle$ and $\langle (b)(e)(d) \rangle$ are both 3-sequences. The sequence $\langle (a, c)(b) \rangle$ is contained in data sequence C1 $\langle_{3(c)} (a, f)_{18} (b)_{31} (a)_{45} (f) \rangle$ because element (a, c) can be contained in the transaction combining $_3(c)$ and $_5(a, f)$ for $5 - 3 \leq 2$ ($swin$). Meanwhile, the $mingap$ and $maxgap$ constraints are satisfied for $18 - 5 \geq 3$ and $18 - 3 \leq 15$. Similarly, $\langle (a, c)(b) \rangle$ is contained in C2. The support of $\langle (a, c)(b) \rangle$ is $2/4$ and it is a time-pattern for $minsup = 50\%$. $\langle (a, c)(b) \rangle$ is also a closed time-pattern since it has no contiguous supersequence with the same support. $\langle (c)(b) \rangle$ is a time-pattern for $\langle (c) \rangle_{sup} = 2/4$ but not closed for its contiguous supersequence $\langle (a, c)(b) \rangle$ having the same support.

Table 1. Example sequence database (DB) and the closed time-constrained sequential patterns.

Sid	Sequence	Closed time-constrained sequential patterns (<i>minsup</i> = 50%, <i>mingap</i> = 3, <i>maxgap</i> = 15, <i>swin</i> = 2)
C1	$\langle_{3_5}(c),_{18_31}(b),_{45}(f)\rangle$	$\langle(a)\rangle:3, \langle(a, c)(b)\rangle:2, \langle(b)\rangle:3, \langle(b)(e)(d)\rangle:2, \langle(c)\rangle:3,$
C2	$\langle_{6_10}(a, c),_{17_24}(b),_{24}(e),_{24}(c, d)\rangle$	$\langle(c, d)\rangle:2, \langle(d)\rangle:3$
C3	$\langle_{1_20}(b),_{27_36}(b, g),_{36}(e),_{36}(d)\rangle$	
C4	$\langle_{5_10}(a),_{21}(d),_{21}(c, d)\rangle$	

The notion of contiguous subsequence (supersequence) is introduced to guarantee the correctness and completeness of closed time-patterns. The subsequence of a (closed) time-pattern is not necessary a (closed) time-pattern if the contiguous relationship is not hold. For example, though $\langle(b)(d)\rangle$ is the subsequence of $\langle(b)(e)(d)\rangle$, time-pattern $\langle(b)(e)(d)\rangle$ does not imply that $\langle(b)(d)\rangle$ is also a time-pattern since $\langle(b)(d)\rangle$ fails the *maxgap* constraint in C3. However, $\langle(b)(e)(d)\rangle$ of support 2/4 ensures that its contiguous subsequences, such as $\langle(b)(e)\rangle$ and $\langle(e)(d)\rangle$, are also time-patterns with support 2/4. That is, having the *maxgap* constraint of 15 days, if pattern $\langle(b)(e)(d)\rangle$ is found, we may infer that (e) could appear after the occurrence of (b), (d) could appear after the occurrence of (e), and both patterns satisfy the *maxgap* constraint. Nevertheless, the immediate occurrence of (d) after (b) cannot be assured with the *maxgap* constraint. The contiguous subsequence notion thus needs to be introduced. Such a definition enables the close time-patterns to have the same expressive power with more compact patterns.

3. CTSP: CLOSE TIME-CONSTRAINED SEQUENTIAL PATTERN MINING

The algorithm we proposed for mining closed time-patterns is called CTSP. CTSP uses the pattern-growth methodology [7, 11, 12] to discover the desired patterns and the bi-directional strategy, similar to BIDE [16], to check the closure property. The algorithm is described as follows.

Definition 1 (frequent item) An item x is called a frequent item in DB if $\langle(x)\rangle.sup \geq minsup$.

Definition 2 (type-1 pattern, type-2 pattern, stem, prefix) Given a frequent pattern P and a frequent item x in DB , P' is a type-1 pattern if it can be formed by adding an item-set consisting of the single item x after the last element of P , and a type-2 pattern if formed by appending x to the last element of P . For type-2 pattern, the lexicographic order of x must be larger than all items in the element. The item x is called the *stem* of the new frequent pattern P' . The prefix pattern (abbreviated as prefix) of P' is P .

For example, $\langle(a)(b)\rangle$ is a type-1 pattern by adding (b) after $\langle(a)\rangle$, and $\langle(a, c)\rangle$ is a type-2 pattern by appending (c) to $\langle(a)\rangle$. Here, (b) and (c) are the stems and $\langle(a)\rangle$ is the prefix pattern.

Definition 3 (last-start time, last-end time, time-index) Let the last element of a frequent pattern P be LE. If ds contains P by having $LE \subseteq e_\gamma \cup e_{\gamma+1} \cup \dots \cup e_\omega$, where $e_\gamma, \dots, e_\omega$ are elements in ds , the occurring time t_γ and t_ω for itemsets e_γ and e_ω are named, respectively, last-start time (abbreviated as lst) and last-end time (abbreviated as let) of P in ds . Every occurrence of the $lst : let$ pair is collected altogether as $[lst_1 : let_1, lst_2 : let_2, \dots, lst_k : let_k]$, $lst_i \leq let_i$ for $1 \leq i \leq k$. Such a timestamp lists is called the *time-index* of P in ds .

In addition to stems, items in the backward direction of a frequent pattern P , *i.e.* occurring before P , may be used to form P' and speed up the mining process. For example, $P = \langle (a)(b) \rangle$ is contained in $s = \langle {}_2(e)_6(a, c)_{10}(b, d)_{18}(a) \rangle$ with time-index [6:10]. Stem (a) (timestamp 18) may extend P to a type-1 pattern $\langle (a)(b)(a) \rangle$ and stem (d) (timestamp 10) may extend P to a type-2 pattern $\langle (a)(b, d) \rangle$ in the forward direction. Considering the (a) in P , item (e) (timestamp 2) and item (c) (timestamp 6) can be used to form contiguous supersequences $\langle (a, c)(b) \rangle$ and $\langle (e)(a)(b) \rangle$, respectively. The two non-stem items are found in the backward direction of P . Thus, we have the following definition.

Definition 4 (extension item, extension period) Given a frequent pattern P contained in ds , a non-stem item α in ds is called an extension item (abbreviated as EI) if it can be used in extending P to form a contiguous supersequence P' satisfying the time constraints. The time periods within which α exists is called extension period (abbreviated as EP). The time period is referred to as backward extension period (abbreviated as BEP) and the item is called backward extension item (abbreviated as BEI). For convenience, we refer to the time period within which stems exist as forward extension period (abbreviated as FEP).

Lemma 1 Given a time-index of frequent pattern P in ds $[lst_1 : let_1, lst_2 : let_2, \dots, lst_k : let_k]$, the FEP satisfies either one of the following conditions: (1) $\exists i, 1 \leq i \leq k, let_i + \text{mingap} \leq \text{FEP} \leq let_i + \text{maxgap}$ (2) $\exists i, 1 \leq i \leq k, let_i - \text{swin} \leq \text{FEP} \leq let_i + \text{swin}$. Fig. 1 illustrates Lemma 1.

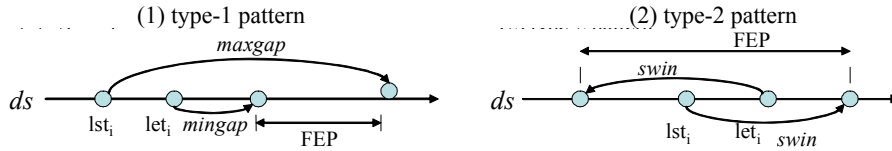


Fig. 1. The forward extension period for (1) type-1 pattern and (2) type-2 pattern.

Definition 5 (timeline) Given a frequent pattern $P = \langle e_1 \dots e_r \dots e_w \rangle$ in ds . If $e_1 \subseteq e_{st1} \cup \dots \cup e_{et1}$, \dots , $e_r \subseteq e_{str} \cup \dots \cup e_{etr}$, \dots , and $e_w \subseteq e_{stw} \cup \dots \cup e_{etw}$, where $e_{st1}, \dots, e_{stw}, e_{et1}, \dots, e_{etw}$ are elements in ds , the list of timestamps $[st1 : et1, \dots, str : etr, \dots, stw : etw]$ where $str \leq etr$ for $1 \leq r \leq w$ is called the *timeline* of P in ds .

Note that the ds may have several timelines of P . For example, $P = \langle (a)(d)(e) \rangle$ can be found in $ds = \langle {}_5(b)_9(a)_{13}(d)_{14}(c)_{16}(e)_{19}(e) \rangle$ with timeline [9:9, 13:13, 16:16] and timeline [9:9, 13:13, 19:19]. The two timelines, implemented as linked lists, are shown in Fig. 2.

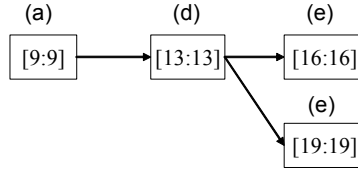


Fig. 2. The timelines of $\langle(a)(d)(e)\rangle$ in $\langle{}_5(b)_9(a)_{13}(d)_{14}(c)_{16}(e)_{19}(e)\rangle$.

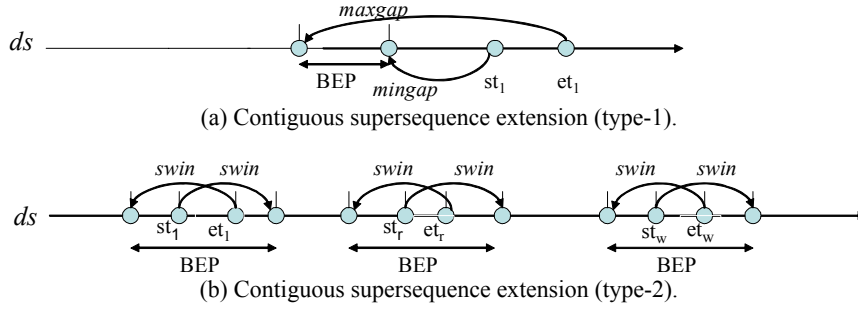


Fig. 3. Two types of backward extension period in extending a contiguous supersequence.

Lemma 2 Given a frequent pattern $P = \langle e_1 \dots e_r \dots e_w \rangle$ contained in ds . For each timeline $[st_1 : et_1, \dots, st_r : et_r, \dots, st_w : et_w]$ of P in ds , the BEP satisfies either one of the following conditions: (1) $et_1 - maxgap \leq BEP \leq st_1 - mingap$; (2) $\exists i, 1 \leq i \leq w, et_i - swin \leq BEP \leq st_i$ or $et_i \leq BEP \leq st_i + swin$. Lemma 2 is illustrated in Fig. 3.

Forming a type-2 pattern with the potential stems using Lemma 1 can be pruned in advance if the stem is lexicographically smaller than the items of the last element in P . Moreover, the time periods are checked on not violating the minimum/maximum gap constraints between adjacent elements when a type-2 pattern is formed.

Lemma 3 Given a frequent pattern P , if a stem to be used in extending P to P' is found in every data sequence containing P , then P is not a closed time-pattern.

Lemma 4 Given a frequent pattern P , if a BEI to be used in extending P to P' is found in every data sequence containing P , then P is not a closed time-pattern.

Fig. 4 outlines the *CTSP* Algorithm, which mines patterns within the pattern-growth framework. The techniques used are similar to the pseudo projection version of *Prefix-span* algorithm [11] and bi-direction closure checking in *BIDE* algorithm [16], while it can handle constraints minimum/maximum gaps and sliding time-window. Assume that the DB can fit into the main memory, *CTSP* first loads DB into memory (as MDB) and scans MDB once to find all frequent items. With respect to each frequent item, *CTSP* then constructs a time-index set for the 1-sequence item and recursively forms time-patterns of longer length. The time-index set is a set of (data-sequence pointer, time-index) pairs. Only those data sequences containing that item would be included. The time-index indicates the list of $lst : let$ pairs as described in Definition 3.

Algorithm: CTSP

Input: DB (a sequence database), $minsup$ (minimum support), $mingap$ (minimum gap), $maxgap$ (maximum gap), $swin$ (sliding time-window).

Output: the set of all closed time-constrained sequential patterns.

1. load DB into memory (as MDB) and scan MDB once to find all frequent items.
2. for each frequent item x ,
 - (1) form the sequential pattern $P = \langle(x)\rangle$.
 - (2) scan MDB once to construct P -Tidx, time-index set of x .
 - (3) call $CMine(P, P$ -Tidx) to mine the closed time patterns.

Subroutine: $CMine(P, P$ -Tidx)

Parameter: P = prefix with support count, P -Tidx = time-index set for P

1. for each data sequence ds in the P -DB, // P -DB: sequences indicated in P -Tidx
 - (1) use Lemma 1 to collect the FEPs of type-1 and type-2 patterns, respectively.
 - (2) for each item in the FEPs of type-1 and type-2 patterns, add one to its support count, respectively.
2. if any support count of a stem is equal to the support count of P , then P is not closed. Otherwise, output P if $Backward(P, P$ -Tidx) return "closed". //Backward checking valid
3. for each item x' found in the FEPs of type-1 pattern and $\langle(x)\rangle.sup \geq minsup$,
 - (1) form the type-1 pattern P' by extending stem x' .
 - (2) use Lemma 1 and the FEPs of each ds in P -DB to construct P' -Tidx, time-index set of x' .
 - (3) call $CMine(P', P'$ -Tidx).
4. for each item x' found in the FEPs of type-2 pattern and $\langle(x)\rangle.sup \geq minsup$,
 - (1) form the type-2 pattern P' by appending stem x' .
 - (2) use Lemma 2 and the FEPs of each ds in P -DB to construct P' -Tidx, time-index set of x' .
 - (3) call $CMine(P', P'$ -Tidx).

Subroutine: $Backward(P, P$ -Tidx)

Parameter: prefix $P = \langle e_1 \dots e_r \dots e_w \rangle$, P -Tidx = time-index set

1. for each element e_j in P
 - (1) for each data sequence ds in the P -DB, // P -DB: sequences indicated in P -Tidx
 - 1.1 find the BEPs of e_j in ds
 - 1.2 add one to the support count of the BEIs
 - (2) if any support count of a BEI is equal to the support count of P , return "not closed"
2. return "closed"

Fig. 4. Algorithm CTSP.

In Fig. 4, $CMine(P, P$ -Tidx) mines type-1 patterns and type-2 patterns having prefix P by effectively locating FEPs with Lemma 1. The P -Tidx is the time-index set for P . $CTSP$ thus never search the data sequences irrelevant to P . Moreover, the FEPs ensure that $CTSP$ locates and counts the effective stems which can form valid patterns, rather than the whole set of items in the data sequence. Furthermore, $CTSP$ adopts forward and backward closure checking to examine the closure of prefix P . The supports of potential

stems and BEIs are handled by forward closure checking and backward closure checking, respectively. If neither stem nor BEI has the same support as P , then P is a desired closed time-pattern. Recursively, for a newly formed type-1 pattern or type-2 pattern P' , its time-index set P' -Tidx is constructed and $\text{CMine}(P', P'$ -Tidx) is invoked. By pushing time attributes deeply into the mining process, CTSP efficiently discovers the desired patterns.

If the database is too large to fit into memory, a projected scheme is applied. The database will be projected to small sub-databases, based on the frequent 1-sequences. Each sub-database then can be mined by the CTSP . A similar technique has been adopted in the Par-CSP algorithm [4]. Therefore, CTSP can mine databases, even when the size of the database is larger than that of main memory.

4. EXPERIMENTAL EVALUATIONS

Extensive experiments were performed on both synthetic and real datasets to assess the performance of the CTSP algorithm. Algorithms GSP and DELISP , which mines all time-constrained sequential patterns without closure checking, were used to compare with CTSP . All experiments were performed on an AMD 2800 + PC with 1GB memory running the Windows XP. The synthetic data was generated using the IBM synthetic dataset generation program [15]. We describe the result of dataset C10T2.5S4I1.25 having 100,000 data sequences ($|DB| = 100k$), with $N_s = 5000$, $N_l = 25000$ and $N = 10000$. The detailed parameters are addressed in [1]. The results of varying $|C|$, $|T|$, $|S|$, and $|I|$ were consistent.

Figs. 5, 6 and 7 show the results of varying *mingap*, *maxgap* and *sliding window* (*swin*) constraints, respectively, on synthetic database C10T2.5S4I1.25N10. The *minsup* was fixed to 0.5%. The number of closed patterns decreases as *mingap* increases or *maxgap* decreases. The gap constraints restrict the generation of patterns so that the running time is decreased. The *swin* relaxes the constraint and allows more patterns to appear so that total execution time is increased. The effect of varying *minsup* from 0.35% to 1% on the same database is shown in Fig. 8.

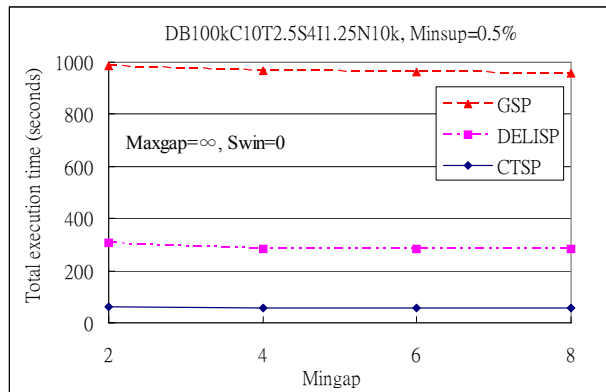


Fig. 5. Results of varying minimum gap.

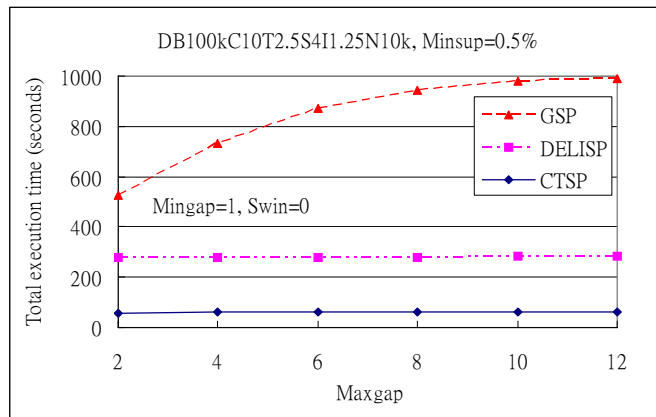


Fig. 6. Results of varying maximum gap.

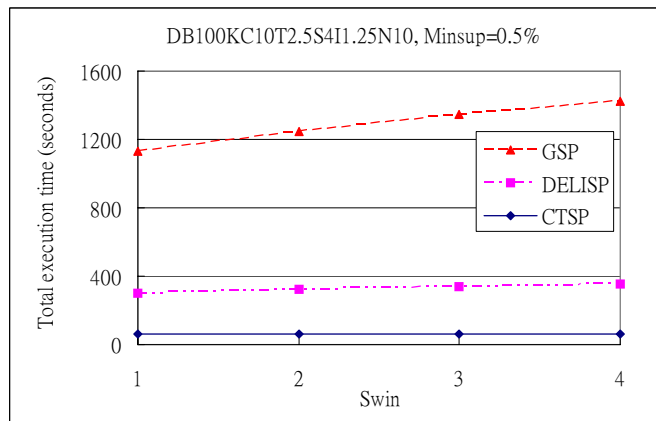


Fig. 7. Results of varying sliding window.

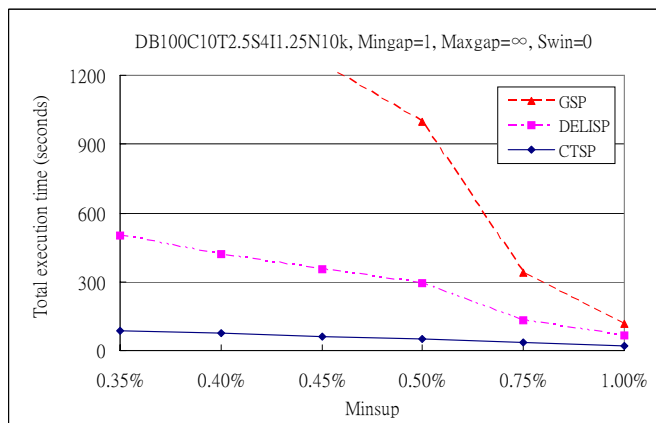


Fig. 8. Results of varying minimum support.

Next, a real database was used in the experiments. The Gazelle database is a click-stream data from Gazelle.com and can be obtained from KDD Cup 2000 [6]. We pre-processed the Gazelle data by using the CookieID as the customer-id to identify distinct customer sequences, using ProductID as the unique item code, and using SessionID to distinguish whether the items are within the same itemset or not. The request times were transformed into appropriate timestamps. In summary, there were 29,369 data sequences, the maximum sequence length and maximum session size were 628 and 267, respectively. The maximal session number in a data sequence was 140. More details can be found in [6].

The *minsup* was varied from 0.04% to 0.06% on the Gazelle database. In Fig. 9, the execution time of *CTSP* was merely 20 seconds when the *minsup* was down to 0.04%. The number of closed time patterns is fewer than the number of non-closed time patterns. In addition, *CTSP* maximizes memory for mining and has fewer number of disk I/O as in *GSP* or *DELISP*. Thus, *CTSP* outperforms *GSP* and *DELISP* with the specified constraints, which lead to a mining of general sequential patterns. Fig. 10 shows the distribution of the discovered closed time-patterns, where the maximum length is up to 14.

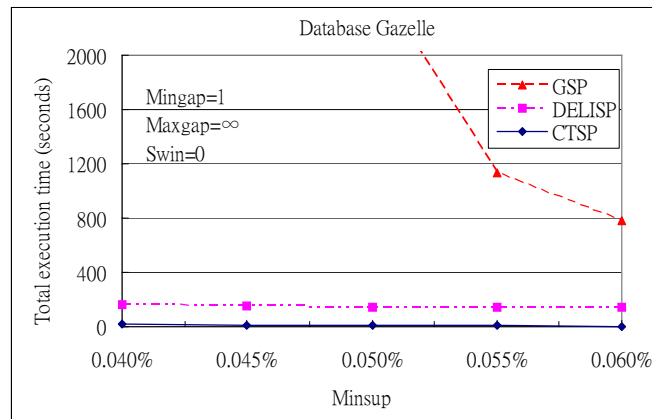


Fig. 9. Results of varying *minsup* with real-dataset *Gazelle* database without constraints.

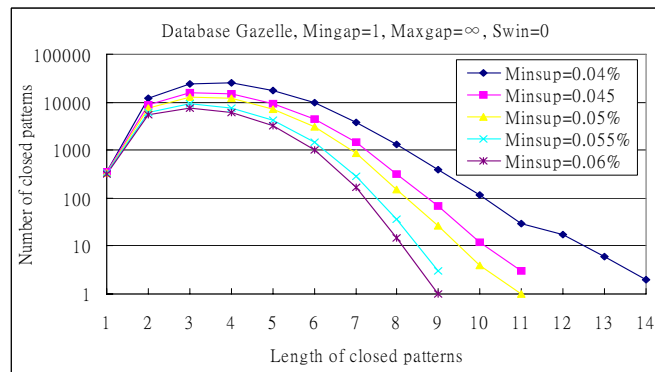


Fig. 10. Distributions of the discovered patterns for *Gazelle* database.

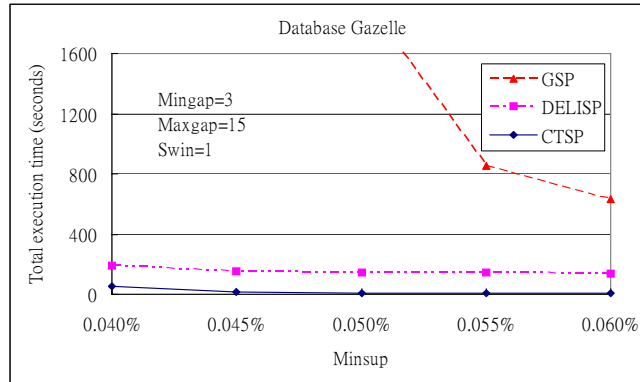


Fig. 11. Results of varying *minsup* with real-dataset *Gazelle* database with constraints.

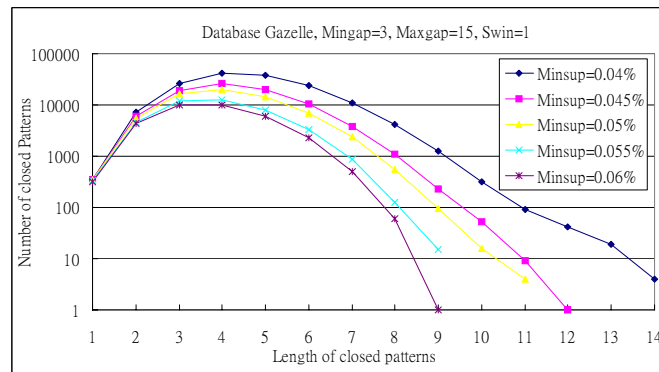


Fig. 12. Distributions of the discovered patterns with time constraints.

Table 2. Total number of discovered patterns.

<i>minsup</i> (%)	0.04	0.045	0.05	0.055	0.06
Time-patterns	515497	153212	101222	53164	40571
Closed Time-patterns	154724	87095	66960	41855	33898

In Fig. 11, the *mingap*, *maxgap* and *sliding window* were set as 3, 15 and 1, respectively. The distribution of the discovered closed time-patterns is shown in Fig. 12. Moreover, Table 2 lists the total number of discovered time-patterns and closed time-patterns. The number of discovered time-patterns is huge because the *Gazelle* is a dense database. All the experimental results show that *CTSP* outperforms *GSP* and *DELISP* in both synthetic and real databases.

To assess the scalability of the algorithm, the number of data sequences was increased from 100K to one million. The result of scaling up databases, as shown in Fig. 13, indicated that *CTSP* has good linear scalability.

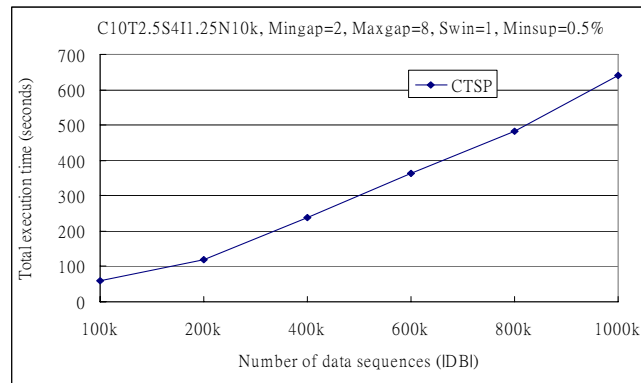


Fig. 13. Linear scalability of the database size.

5. CONCLUSION

In this paper, we have presented an efficient algorithm called *CTSP* for mining closed sequential patterns with minimum/maximum gap and sliding window constraints. The contributions of the proposed algorithm are as follows. First, we define the closure of closed time-patterns. Many algorithms were proposed to discover closed patterns but their definitions of closure are no longer suitable for the time-constrained patterns. Our definition of closure is based on the contiguous supersequence so that the accuracy and completeness of closed time-constrained sequential patterns can be ensured. Second, to the best of our knowledge, no algorithm has been proposed for mining closed sequential patterns with time constraints. The *CTSP* algorithm is the first algorithm that successfully mines closed time-constrained sequential patterns. *CTSP* uses memory-indexes and the time constraints to shrink the search-space effectively within the pattern-growth framework. The closure checking is bi-directionally performed. The experimental results show that *CTSP* has good performance with gap constraints, both for synthetic and real datasets.

REFERENCES

1. R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of the 11th International Conference on Data Engineering*, 1995, pp. 3-14.
2. J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bit-map representation," in *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 429-435.
3. D. Y. Chiu, Y. H. Wu, and A. L. P. Chen, "An efficient algorithm for mining frequent sequences by a new strategy without support counting," in *Proceedings of the 20th International Conference on Data Engineering*, 2004, pp. 375-386.
4. S. Cong, J. Han, and D. A. Padua, "Parallel mining of closed sequential patterns," in *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Databases*, 2005, pp. 562-567.
5. M. N. Garofalakis, R. Rastogi, and K. Shim, "SPIRIT: sequential pattern mining

- with regular expression constraints,” in *Proceedings of the 25th International Conference on Very Large Data Bases*, 1999, pp. 223-234.
6. R. Kohavi, C. Brodley, B. Frasca, L. Mason, and Z. Zheng, “KDD-Cup 2000 organizers’ report: Peeling the onion,” *SIGKDD Explorations*, Vol. 2, 2000, pp. 86-98.
 7. M. Y. Lin and S. Y. Lee, “Fast discovery of sequential patterns through memory indexing and database partitioning,” *Journal of Information Science and Engineering*, Vol. 21, 2005, pp. 109-128.
 8. M. Y. Lin and S. Y. Lee, “Efficient mining of sequential patterns with time constraints by delimited pattern-growth,” *Knowledge and Information Systems*, Vol. 7, 2005, pp. 499-514.
 9. F. Masseglia, P. Poncelet, and M. Teisseire, “Pre-processing time constraints for efficiently mining generalized sequential patterns,” in *Proceedings of the 11th International Symposium on Temporal Representation and Reasoning*, 2004, pp. 487-495.
 10. S. Orlando, R. Perego, and C. Silvestri, “A new algorithm for gap constrained sequence mining,” in *Proceedings of the ACM Symposium on Applied Computing*, 2004, pp. 540-547.
 11. J. Pei, J. Han, B. Moryazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu, “PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth,” in *Proceedings of the 17th International Conference on Data Engineering*, 2001, pp. 215-224.
 12. J. Pei, J. Han, and W. Wang, “Mining sequential patterns with constraints in large databases,” in *Proceedings of the 11th International Conference on Information and Knowledge Management*, 2002, pp. 18-25.
 13. T. Petre, X. Yan, and J. Han, “TSP: mining top- k closed sequential patterns,” *Knowledge and Information Systems*, Vol. 7, 2005, pp. 438-457.
 14. M. Seno and G. Karypis, “SLPMiner: an algorithm for finding frequent sequential patterns using length-decreasing support constraint,” in *Proceedings of the IEEE International Conference on Data Mining*, 2002, pp. 418-425.
 15. R. Srikant and R. Agrawal, “Mining sequential patterns: generalizations and performance improvements,” in *Proceedings of the 5th International Conference on Extending Database Technology*, 1996, pp. 3-17.
 16. J. Wang and J. Han, “BIDE: efficient mining of frequent closed sequences,” in *Proceedings of the 20th International Conference on Data Engineering*, 2004, pp. 79-90.
 17. X. Yan, J. Han, and R. Afshar, “CloSpan: mining closed sequential patterns in large databases,” in *Proceedings of the 3rd SIAM International Conference on Data Mining*, 2003, pp. 166-177.
 18. M. J. Zaki, “SPADE: an efficient algorithm for mining frequent sequences,” *Machine Learning Journal*, Vol. 42, 2001, pp. 31-60.
 19. M. J. Zaki, “Sequence mining in categorical domains: incorporating constraints,” in *Proceedings of the 9th International Conference on Information and Knowledge Management*, 2000, pp. 422-429.



Ming-Yen Lin (林明言) received the B.S. (1988), the M.S. (1990), and the Ph.D. (2003) degrees in Computer Science and Information Engineering, all from Department of Computer Science and Information Engineering in National Chiao Tung University, Taiwan. He is an assistant professor in Department of Information Engineering and Computer Science in Feng Chia University, Taiwan. His research interests include data mining, data stream management systems, and bioinformatics.



Sue-Chen Hsueh (薛夙珍) received the B.S. degree (1990) in Management Science, the M. S. degree (1994) and the Ph.D. degree (1998) in Information Management, from National Chiao Tung University, Taiwan. She is an assistant professor in Department of Information Management, Chaoyang University of Technology, Taiwan. Her research interests include information security, electronic payment systems, and mobile mining.



Chia-Wen Chang (張家汶) was born in Taiwan, R.O.C., 1981. He received the B.S. degree from Department of Computer Science and Information Engineering, Chung Hua University in 2004 and the M.S. degree from Feng Chia University in 2006, Taiwan. His research interests include mining sequential patterns with time constraints, multimedia data mining and management, and computer graphics.