KHAI THÁC K MẪU TUẦN TỰ ĐÓNG

Trần Phước Nghĩa
Trường Đại học Bạc Liêu
Email: tpnghia@blu.edu.vn.

Tóm tắt: *Khai thác mẫu tuần tự là một phần quan trọng của khai thác dữ liệu với các ứng dụng rộng rãi. Tuy nhiên, việc tùy chỉnh thông số minsup để phù hợp trong các thuật toán khai thác mẫu tuần tự nhằm tạo ra đúng số mẫu mà người dùng mong muốn là điều rất khó khăn và tốn thời gian. Để giải quyết vấn đề này, thuật toán khai thác k mẫu tuần tự đóng TSP đưa ra phương án giới hạn lại số lượng k mẫu cần khai thác, nhưng thời gian thực hiện và bộ nhớ sử dụng của thuật toán cao. Vì thế, bài viết này đề xuất thuật toán TKCS tìm k mẫu tuần tự đóng dựa trên thuật toán TKS[2]. Với k mẫu nhập vào thuật toán sẽ trả về k mẫu có độ hỗ trợ cao nhất trong cơ sở dữ liệu (CSDL). Kết quả thực thi cho thấy, Thuật toán TKCS có hiệu suất tốt hơn rất nhiều so với thuật toán TSP về chi phí thời gian cũng như bộ nhớ sử dụng. Ngoài ta thuật toán TKCS còn xử lý tốt trên các CSDL khác nhau, đặc biệt là các CSDL với mẫu lớn.*
Từ khóa: *Khai thác dữ liệu, mẫu tuần tự, mẫu tuần tự đóng, cơ sở dữ liệu chuỗi.*
Nhận bài: 15/01/2026; Biên tập: 16/01/2026; Phản biện: 19/01/2026; Duyệt đăng: 26/01/2026.

1. Giới thiệu

Khai thác dữ liệu là lĩnh vực đã và đang được nghiên cứu trong nhiều năm qua với mục đích hỗ trợ các nhà quản lý tìm ra mối quan hệ giữa các sản phẩm trong số lượng lớn danh mục sản phẩm và nhờ đó có thể giúp tăng doanh thu. Quá trình khai thác dữ liệu là quá trình phát hiện ra các mẫu thông tin có giá trị tiềm ẩn trong CSDL.

Khai thác luật kết hợp [1] là một trong những phương thức hay và phổ biến nhất để đạt được mục đích này. Việc khai thác các luật kết hợp nhằm mục đích phát hiện ra các mối quan hệ giữa các tập thuộc tính trong CSDL với nhau, trong đó khai thác tập phổ biến đóng vai trò quan trọng trong việc khai thác các luật kết hợp. Sự đa dạng và phong phú của dữ liệu hình thành nên nhiều loại dữ liệu khác nhau: dữ liệu giao tác (transaction), dữ liệu chuỗi (sequence), thời gian (time)… Trước tình hình đó, việc khai thác và chọn lọc những dữ liệu có ích từ lượng dữ liệu đó là việc cần thiết, đóng vai trò quyết định thành công trong mọi hoạt động. Các dữ liệu được chắt lọc đó giúp cải thiện hoạt động trong hiện tại hay đưa ra những dự đoán giúp việc đưa ra quyết định trong tương lai chính xác hơn.

Trên thực tế, mỗi một sản phẩm mà khách hàng mua lại có thể có giá khác nhau. Tương tự mỗi hạng mục trong giao dịch cũng có các trọng số khác nhau tùy từng loại CSDL cụ thể. Chính vì vậy, nhiều nghiên cứu đã được thực hiện, nhiều thuật toán được đề xuất trong lĩnh vực này. Trong đó, thuật toán TKS (Top-K Sequential pattern mining) [2] được đánh giá cao bởi chi phí thực hiện thấp hơn nhiều lần so với các thuật toán khác trong việc khai thác k mẫu tuần tự phổ biến. Dựa vào đó để làm nền tảng tiến hành nghiên cứu bài toán Khai Thác Top K mẫu tuần tự đóng.

2. Nội dung nghiên cứu

2.1. Các khái niệm cơ bản

2.1.1. Khái niệm về chuỗi dữ liệu.

Cho I = $\{i_1, i_2, \ldots, i_k\}$ là một tập các item. Tập con của I gọi là Itemset. Chuỗi $s = \langle t_1, t_2, \ldots, t_m \rangle$ $(t_i \subseteq I)$ là một danh sách có thứ tự. Giả sử các item trong mỗi itemset được nhóm theo thứ tự.

Ví dụ: Xét CSDL như bảng 1

Chuỗi *s*1 có 5 itemset xảy ra theo thứ tự <(a)(abc)(ac)(d)(cf)>. Chiều dài của *s*, *l(s)* là tổng số các item trong s còn được gọi là *l-sequence*.

Ví dụ: Chuỗi <(ac)(d)> là một 3-sequence có kích thước là 2.

Chuỗi $\alpha = \langle a_1, a_2, \ldots, a_m \rangle$ là một chuỗi con của chuỗi khác $\beta = \langle b_1, b_2, \ldots, b_n \rangle$ kí hiệu là $\alpha \subseteq \beta$ nếu và chỉ nếu $\exists i_1, i_2, \ldots i_m$, sao cho $1 \le i_1 < i_2 < \ldots < i_m \le n$ và $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, ... $a_m \subseteq b_{im}$. Chúng gọi β là chuỗi cha của α.

Ví dụ: Chuỗi <(a)(d)> là chuỗi con của <(ad)(c)(bc)(ae)> nhưng <(a)(d)> không phải là chuỗi con của chuỗi <(ad)> và ngược lại.

Bảng 1: CSDL chuỗi

SID	Sequences
1	<(a)(abc)(ac)(d)(cf)>
2	<(ad)(bc)(c)(ae)>
3	<(ef)(ab)(df)(c)(b)>
4	<(ae)(af)(c)(b)(c)>
5	<(ce)(bf)(ad)(d)(e)(ac)>

** Cơ sở dữ liệu chuỗi:*

Cơ sở dữ liệu chuỗi $D = \{s_1, s_2, \ldots, s_n\}$ là một tập các chuỗi có dạng *(SID,s)*, trong đó *SID* là định danh của chuỗi và *s* là chuỗi các itemset.

Ví dụ: Xét CSDL chuỗi như sau:

** Độ hỗ trợ:*

Xét CSDL chuỗi *D*, mỗi chuỗi có một chỉ số định danh duy nhất. Độ hỗ trợ tuyệt đối của một mẫu tuần tự α là tổng số chuỗi trong *D* có chứa p, ký hiệu $sup_D(p) = |\{s \mid s \in D \text{ và } \alpha \subseteq s\}|$. Độ hỗ trợ tương

đối của *p* là tỉ lệ phần trăm chuỗi trong *D* chứa *p*. Ở đây, mức hỗ trợ tuyệt đối hoặc tương đối sẽ được sử dụng chuyển đổi qua lại, kí hiệu là *sup(p).*

Ví dụ: Xét CSDL bảng 1. Chuỗi p = <(a)(d)> xuất hiện trong chuỗi s_1, s_2, s_3, s_5. Vậy độ hỗ trợ của chuỗi p là 4

** Mẫu:*

Mẫu là một chuỗi con của một chuỗi dữ liệu. Mỗi itemset trong một mẫu còn được gọi là một thành phần (element).

Ví dụ: Mẫu α = <(ab)> là chuỗi con của chuỗi s_1

** Mẫu tuần tự:*

Cho ngưỡng hỗ trợ tối thiểu (*minsup*) xác định bởi người dùng, *minsup* $\in$ (0, 1]. Một mẫu α được coi là phổ biến nếu độ hỗ trợ của nó lớn hơn hoặc bằng *minsup*: *sup*(α) ≥ *minsup*, khi đó α được gọi là mẫu tuần tự.

2.1.2. Khai thác mẫu tuần tự

Cho trước CSDL chuỗi và ngưỡng *minsup*. Khai thác mẫu tuần tự là đi tìm tập đầy đủ tất cả các mẫu tuần tự có trong CSDL và có độ lớn hơn hoặc bằng ngưỡng minsup đã cho.

Có hai dạng tổ chức dữ liệu cơ bản:

- Dạng biểu diễn ngang: Dữ liệu được tổ chức theo chiều ngang, mỗi hàng đại diện cho dãy sự kiện (event) tương ứng với đối tượng (object).

- Dạng biểu diễn dọc: Dữ liệu được tổ chức theo chiều dọc, mỗi bảng trình bày cho một item và chỉ ra danh sách của các chuỗi mà item xuất hiện và vị trí mà item đó xuất hiện.

Ví dụ**:** Xét CSDL như bảng 1.

CSDL trên có thể biểu diễn theo 2 cách sau:

Bảng 2. *Biểu diễn ngang*

SID	Sequences
1	<(a)(abc)(ac)(d)(cf)>
2	<(ad)(bc)(c)(ae)>
3	<(ef)(ab)(df)(c)(b)>
4	<(ae)(af)(c)(b)(c)>
5	<(ce)(bf)(ad)(d)(e)(ac)>

Bảng 3. Biểu diễn dọc.

Sequenc-es	SID
a	1,2,3,4,5
b	1,2,3,4,5
c	1,2,3,4,5
d	1,2,3,5
e	2,3,4,5
f	1,3,4,5

Trong hai cách tổ chức dữ liệu theo chiều dọc và theo chiều ngang, thao tác đếm độ hỗ trợ cho một sự kiện ở CSDL được tổ chức theo chiều dọc đơn giản và nhanh hơn. Bởi vì theo cách tổ chức này, có thể lấy được ngay các đối tượng ứng với sự kiện mà không phải duyệt toàn bộ CSDL. Hơn nữa, đối với CSDL lớn, việc tổ chức theo chiều dọc mang tính cô đọng, giúp thực thi nhanh hơn và cho phép lặp lại việc tìm các mẫu tuần tự một cách dễ dàng. Tuy nhiên, dữ liệu gốc ban đầu thường được tổ chức theo chiều ngang, nếu muốn tổ chức theo chiều dọc phải có bước tiền xử lý để chuyển đổi.

2.3. Các thuật toán khai thác mẫu tuần tự

Bài toán khai thác mẫu tuần tự được đề xuất đầu tiên bởi Agrawal và Srikant vào năm 1995. AprioriAll [3] là thuật toán đầu tiên được thiết kế để giải quyết bài toán khai thác mẫu tuần tự trên CSDL chuỗi giao dịch. AprioriAll dựa trên thuật toán khai thác mẫu phổ biến Apriori [4], là thuật toán nền tảng làm cơ sở cho các thuật toán về sau.

** Thuật toán AprioriAll:*

Để tìm mẫu tuần tự, thuật toán AprioriAll [3] gồm 3 giai đoạn chính: Tìm itemset phổ biến, biến đổi CSDL và tìm mẫu tuần tự trên dữ liệu đã biến đổi. Ở giai đoạn 1, thuật toán tiến hành duyệt toàn bộ CSDL ban đầu để tìm các itemset phổ biến. Sau đó, ánh xạ tập itemset phổ biến tìm được sang tập số nguyên. Việc ánh xạ nhằm mục đích coi một temset phổ biến là một thực thể riêng biệt và thời gian so sánh hai itemset phổ biến bất kỳ đều như nhau. Hơn nữa, giúp giảm thời gian kiểm tra một chuỗi có là chuỗi con của chuỗi dữ liệu trong CSDL ban đầu không.

Giai đoạn 2, trong CSDL chuỗi ban đầu, mỗi chuỗi được thay thế bởi tập tất cả các itemset phổ biến có chứa trong chuỗi đó. Nếu itemset không chứa itemset con phổ biến nào thì loại bỏ itemset đó khỏi tập chuỗi dữ liệu. Nếu chuỗi dữ liệu không chứa itemset phổ biến nào thì loại bỏ chuỗi đó khỏi CSDL. Sau khi biến đổi, mỗi chuỗi sẽ được đại diện bởi một dãy các itemset phổ biến.

Giai đoạn 3, tìm mẫu tuần tự dựa trên kết quả từ giai đoạn tìm itemset phổ biến, ta có được tập các mẫu tuần tự có kích thước là 1. Giải thuật dựa trên nguyên tắc Apriori: Mọi tập con của tập phổ biến phải là tập phổ biến, mọi tập cha của tập không phổ biến đều không phổ biến. Tập ứng viên gồm các mẫu độ dài k được phát sinh bằng cách kết các mẫu độ dài *(k-1)*, sau đó dựa vào nguyên tắc Apriori và *minsup* để loại bỏ các mẫu không phổ biến.

Như vậy, để tìm được mẫu tuần tự, giải thuật AprioriAll phải phát sinh các ứng viên, nhưng số lượng ứng viên tạo ra rất lớn dễ dẫn đến tình trạng "nghẽn cổ chai". Với chuỗi độ dài n thì số ứng viên có thể tạo ra là do đó không đủ bộ nhớ để xử lý. Mặt khác, để tìm tất cả các mẫu tuần tự, thuật toán AprioriAll phải duyệt CSDL nhiều lần vì với mỗi tập ứng viên để đếm độ hỗ trợ phải duyệt toàn bộ CSDL. Đối với bài toán khai thác mẫu tuần tự, các yếu tố ảnh hưởng đến tính hiệu quả của thuật toán là cách thức tổ chức dữ liệu và thuật toán giải quyết.

Do đó, phải sử dụng cấu trúc dữ liệu thích hợp và thuật toán tối ưu. Như vậy, các đặc trưng ảnh hưởng đến tốc độ thực thi là cách tổ chức biểu diễn dữ liệu để lưu trữ vào bộ nhớ, cách duyệt dữ liệu để xử lý, các chiến lược tìm kiếm. Ngoài ra, sử dụng một số đặc trưng khác như vận dụng lý thuyết đồ thị, đưa ra những ràng buộc cho bài toán sẽ giúp thuật toán thực thi nhanh hơn, các mẫu tuần tự tìm được có giá trị hơn. Chính vì vậy, tiếp cận theo

nhiều hướng khác nhau, xuất phát từ thuật toán nền tảng AprioriAll, các nhóm nghiên cứu đã đưa ra nhiều thuật toán khác nhau để giải quyết bài toán khai thác mẫu tuần tự.

2.2. Khai thác top k mẫu tuần tự đóng

Khai thác mẫu tuần tự đã được nghiên cứu rộng rãi trong cộng đồng khai thác dữ liệu và có rất nhiều ứng dụng trong thực tế. Khai thác mẫu tuần tự là tìm tất cả các chuỗi con chung (subsequences) xuất hiện nhiều hơn minsup lần trong các chuỗi (sequences) của CSDL, với minsup là ngưỡng hỗ trợ tối thiểu do người dùng định nghĩa [5]. Cho đến nay, mặc dù nhiều nghiên cứu để thiết kế ra các thuật toán khai thác mẫu tuần tự đã được thực hiện nhưng một vấn đề quan trọng là làm cách nào để người sử dụng có thể chọn được ngưỡng minsup nhằm tạo ra một số lượng mong muốn các mẫu. Tùy thuộc vào sự lựa chọn ngưỡng minsup, thuật toán có thể trở nên rất chậm và tạo ra một số lượng rất lớn các kết quả hoặc quá ít kết quả hoặc không có kết quả nào, bỏ qua các thông tin có giá trị. Vấn đề này rất quan trọng bởi vì trong thực tế, người sử dụng chỉ có nguồn tài nguyên giới hạn (thời gian và không gian lưu trữ) nên không thể phân tích quá nhiều mẫu kết quả và việc tinh chỉnh thông số minsup là rất tốn thời gian.

Để giải quyết vấn đề này, người ta đã đề xuất xác định lại vấn đề của khai thác các mẫu tuần tự như là khai thác k mẫu tuần tự phổ biến, với k là số mẫu tuần tự đã tìm ra và được định nghĩa bởi người sử dụng.

Tuy nhiên, để tìm ra các mẫu tuần tự phổ biến với k mẫu cho trước thì chỉ có thuật toán TSP [7] triển khai, nhưng vẫn còn nhiều bất cập khi khai thác nhất là về mặt thời gian và bộ nhớ. Chúng tôi đề xuất thuật toán TKCS để tìm ra tập đóng với thời gian được rút ngắn và bộ nhớ giảm.

2.2.1. Thuật toán TKS

Khai thác mẫu tuần tự là một phần quan trọng của khai thác dữ liệu với các ứng dụng rộng rãi. Tuy nhiên, việc tinh chỉnh thông số minsup trong các thuật toán khai thác mẫu tuần tự để tạo ra đủ số mẫu mong muốn là rất khó khăn và tốn thời gian.

Để giải quyết vấn dề này, người ta đã đề xuất xác định lại vấn đề khai thác mẫu tuần tự như là khai thác k mẫu tuần tự phổ biến, với k là số mẫu tuần tự được tìm ra (được trả về) và được thiết lập bởi người dùng. Thuật toán tốt nhất hiện nay để giải quyết vấn đề này là TKS (Top-K Sequential pattern mining)[5].

Thuật toán TKS sử dụng CSDL bitmap dọc để trình bày dữ liệu và sử dụng thủ tục tạo ứng viên căn bản của SPAM[8] để dò tìm và mở rộng các mẫu, đồng thời áp dụng một vài chiến lược để tăng hiệu quả khai thác k mẫu tuần tự phổ biến.

** Cơ sở dữ liệu bitmap dọc*

Cho CSDL chuỗi D chứa q item và m chuỗi (sequence), (sequence), size(i) là số itemset trong chuỗi thứ i. CSDL bitmap dọc của D, ký hiệu V (D) được định nghĩa như là một tập của q bit vector có kích thước , sao cho:

Mỗi item x có một bit vector tương ứng.

Nếu item x xuất hiện trong itemset thứ p của chuỗi thứ t trong D thì bit thứ của bit vector bv(x) được gán là 1, ngược lại là 0.

Ví dụ: Bảng bên dưới trình bày các bit vector được xây dựng cho mỗi item từ CSDL được cho trong bảng 4.

Bảng 4. CSDL bitmap dọc được xây dựng từ CSDL

Item Bit vector	Item Bit vector
A	111001001010001100000100 1
B	010000100010010001001 0000
C	011010110000100010110000 1
D	000101000001000000000110 0
E	000000001100001000010001 0
F	000010000101000100001000 0

* Phương thức tạo ứng viên trong thuật toán SPAM

Thủ tục tạo ứng viên được trình bày trong hình 1 bên dưới. Thủ tục này nhận hai tham số đầu vào là CSDL chuỗi và ngưỡng minsup. Các bước thực hiện như sau:

SPAM(*CSDL chuỗi D, minsup)*
1. Quét CSDL để tạo *V(D)* và xác định S_{init} (danh sách các items phổ biến).
2. **FOR** each item $s \in S_{init}$,
3. **SEARCH**(⟨*s*⟩, S_{init}, tập các items từ S_{init} có thứ tự từ điển lớn hơn *s*,

Hình 1. Thuật toán SPAM

SEARCH(*pat*, S_n, I_n, *minsup*)
1. Đầu ra: mẫu *pat*.
2. $S_{temp} := I_{temp} := \varnothing$
3. **FOR** each item $i \in S_n$,
4. **IF** mở rộng *s-extension* của *pat* là phổ biến **THEN** $S_{temp} := S_{temp} \cup \{i\}$.
5. **FOR** each item $j \in S_{temp}$,
6. **SEARCH**(mở rộng *s-extension* của *pat* với *j*, S_{temp} , các phần tử trong S_{temp} lớn hơn *j, minsup*).
7. **FOR** each item $i \in I_n$,
8. **IF** mở rộng *i-extension* của *pat* là phổ biến **THEN** $I_{temp} := I_{temp} \cup \{i\}$.
9. **FOR** each item $j \in I_{temp}$,
10. **SEARCH**(mở rộng *i-extension* của *pat* với *j*, S_{temp} , tất cả các

Hình 2. Thủ tục tạo ứng viên trong thuật toán SPAM

- Quét CSDL một lần để xây dựng CSDL bitmap dọc (xem bảng 4), đồng thời đếm độ hỗ trợ của các item đơn.

- Với mỗi item phổ biến *s*, nó gọi thủ tục "SEARCH". Thủ tục này xuất ra mẫu ⟨{*s*}⟩ và đệ quy dò tìm các mẫu ứng viên bắt đầu bằng tiền tố ⟨{*s*}⟩.

- Thủ tục SEARCH (hình 3.2) nhận bốn tham số đầu vào là: một mẫu tuần tự *pat*, hai tập các items S_n, I_n được dùng để nối thêm vào mẫu *pat* nhằm tạo các ứng viên và ngưỡng *minsup*. S_n là tập các

tính xách tay Sony Vaio có cấu hình CPU Intel core i5-7200, 16G RAM và sử dụng hệ điều hành Microsoft Windows 10, cài đặt trên ngôn ngữ lập trình Java.

** Bộ dữ liệu chạy thực nghiệm*

Bảng 4. Bộ CSDL chạy thực nghiệm và các thuộc tính liên quan

DATASET SE-QUENCE	COUNT DIS-TINCT	ITEM COUNT	AVG. SEQ. LENGTH (ITEMS)	TYPE OF DA-TA
Leviathan	5834	9025	33.81	book
Bible	36369	13905	21.64	book
Sign	800	267	51.99	sign language utter-ances
FIFA	20450	2990	34.74	web click stream
BmsWebView1	59601	497	2.42	web click stream
BmsWebView2	77512	3340	4.62	web click stream

**Kết quả thực nghiệm giữa TKCS và TSP*

Chạy thực nghiệm 2 thuật toán TKCS và TSP với bộ dữ liệu Sign và Leviathan, Bible, FIFA, BmsWebView1, BmsWebView2 lần lượt cho các mẫu k = 50, 100 , 200, 300, 400 ta thu về được bảng kết quả như sau:

Bảng 5. Kết quả thực nghiệm TKCS và TSP trên các bộ dữ liệu

CSDL	Số lượng mẫu k	Bộ nhớ sử dụng (MB)	
		TKCS	TSP
Sign	50	109	281
	100	163	285
	200	240	290
	300	250	293
	400	260	300
Leviathan	50	250	318
	100	320	384
	200	435	522
	300	590	668
	400	680	906
Bible	50	363	574
	100	399	695
	200	450	872
	300	550	895
	400	661	950
FIFA	50	278	560
	100	429	743
	200	556	789
	300	630	851
	400	759	935
BmsWebView1	50	72	183
	100	129	298
	200	178	319
	300	219	368
	400	293	422
BmsWebView2	50	206	273
	100	253	336
	200	350	411
	300	412	493
	400	550	583

3. Kết luận

Cơ sở lý thuyết khai thác mẫu tuần tự và mẫu tuần tự đóng giúp chúng ta thấy được tầm quan trọng trong khai thác mẫu tuần tự đóng hiện nay. Khai thác mẫu tuần tự đóng là tìm tất cả các chuỗi cha, loại bỏ các chuỗi con có cùng độ hỗ trợ, ứng với k mẫu nhập vào. Tìm ra những mẫu có độ hỗ trợ cao nhất và loại bỏ các trường hợp tìm ra các mẫu bị trùng lắp. Cho đến nay, rất nhiều thuật toán được đưa ra trong đó có thuật toán TSP đã giải quyết được vấn đề trên nhưng khi thực thi thuật toán còn nhiều hạn chế về tốc độ cũng như dung lượng lưu trữ. Để giải quyết những khuyết điểm trên, chúng tôi đề xuất triển khai thuật toán TKCS. Với những ưu thế từ thuật toán gốc TKS mang lại, thêm vào đó là các thuật giải mới nên thuật toán TKCS đã có những bước tiến vượt trội hơn so với TSP ■

Tài liệu tham khảo

[1]. Fournier-Viger, A Gomariz, T Gueniche, E Mwamikazi, R Thomas (2013). *International Conference on Advanced Data Mining and Applications*.109- 120.

[2]. Agrawal and R. Srikant (1995). *"Mining sequential patterns," Proc. 11th Int*. Conf. Data Eng., pp. 3 - 14.

[3]. Agrawal, T. Imieliński, and A. Swami (1993). "*Mining association rules between sets of items in large databases".* ACM SIGMOD Rec., vol. 22, no. 2, pp. 207 – 216.

[4]. Srikant and R. Agrawal (1996). *"Mining Sequential Patterns: Generalizations and Performance Improvements," Proc.* 5th Int. Conf. Extending Database Technol. Adv. Database Technol., pp. 3 - 17.

Mining K closed sequential patterns

Trần Phước Nghĩa

Bac Lieu University - Email: tpnghia@blu.edu.vn.

Abstract: *Sequential pattern mining is an important area of data mining with wide-ranging applications. However, adjusting the minimum support (minsup) parameter appropriately in sequential pattern mining algorithms to generate exactly the number of patterns desired by users is both difficult and time-consuming. To address this issue, the TSP algorithm for mining k closed sequential patterns was proposed to limit the number of patterns to k. Nevertheless, this algorithm requires high execution time and memory consumption. Therefore, this paper proposes the TKCS algorithm to mine k closed sequential patterns based on the TKS algorithm [2]. Given an input value k, the algorithm returns the top k patterns with the highest support in the database. Experimental results show that the TKCS algorithm significantly outperforms the TSP algorithm in terms of both execution time and memory usage. In addition, the TKCS algorithm performs well across different databases, especially those containing large patterns.*

Keywords: *Data mining, sequential patterns, closed sequential patterns, sequence database.*