TẠP CHÍ KHOA HỌC
TRƯỜNG ĐẠI HỌC SƯ PHẠM TP HỒ CHÍ MINH
Tập 22, Số 10 (2025): 1887-1899
Website: https://journal.hcmue.edu.vn
ISSN: 2734-9918

HO CHI MINH CITY UNIVERSITY OF EDUCATION
JOURNAL OF SCIENCE
Vol. 22, No.10 (2025): 1887-1899
https://doi.org/10.54607/hcmue.js.22.10.5056(2025)

Bài báo nghiên cứu

KHAI THÁC CÁC MẪU PHỔ BIẾN TRÊN CƠ SỞ DỮ LIỆU TRỌNG SỐ ĐỘNG

Ngô Việt Anh, Nguyễn Duy Hàm*

Trường Đại học Sài Gòn, Việt Nam

**Corresponding author: Nguyễn Duy Hàm – Email: ndham@sgu.edu.vn*

Ngày nhận bài: 11-6-2025; Ngày nhận bài sửa: 08-9-2025; Ngày duyệt đăng: 16-9-2025

TÓM TẮT

Khai thác các mẫu là một trong những bài toán cơ bản của khai thác dữ liệu hiện đại. Trong đó khai thác mẫu trên các loại cơ sở dữ liệu có định lượng (Frequent weighted patterns -PWPs) là một bài toán quan trọng của khai thác mẫu, đây là bài toán nhằm tìm ra các mẫu phổ biến trên cơ sở dữ liệu (CSDL) định lượng. Tuy nhiên các nghiên cứu hiện tại chưa quan tâm đến các CSDL định lượng có sự thay đổi trọng số của các mục (dynamic weighted Database - dWDB). Trong thực tế, nhiều CSDL mà trọng số của các mục có thể thay đổi theo thời gian, khi trọng số của các mục đại diện cho mức độ quan trọng như lợi nhuận của các mặt hàng hay mức độ quan trọng của các mặt hàng trong từng thời điểm nhất định (Ví dụ máy lạnh sẽ được bán nhiều vào mùa hè, khẩu trang y tế sẽ có vai trò quan trọng trong các đợt dịch thông qua đường hô hấp...). Trong bài báo này, trước hết chúng tôi giới thiệu một bài toán mới về khai thác PWPs với các mục có trọng số động từ cơ sở dữ liệu định lượng - CSDL định lượng động. Tiếp theo, một thuật toán gọi là dFWNL được phát triển sử dụng cấu trúc dữ liệu mới là dWNList để khai thác PWPs từ dWDB. Cuối cùng, chúng tôi thực hiện thực nghiệm trên nhiều dWDB khác nhau để chứng minh hiệu quả của các thuật toán đề xuất.

Từ khoá: CSDL trọng số động; mẫu phổ biến; cấu trúc WNlist; CSDL trọng số

1. Giới thiệu

1.1. Tổng quan về bài toán Khai thác mẫu phổ biến

Khai thác mẫu phổ biến (Frequent Pattern Mining) lần đầu tiên được Agrawal và cộng sự (Agrawal et al., 1993) giới thiệu và đã trở thành một vấn đề quan trọng trong khai thác dữ liệu. Có nhiều thuật toán hiện có để Khai thác mẫu phổ biến một cách hiệu quả như Apriori (Agrawal et al., 1993; Agrawal & Srikant, 1994), FP-Growth (Han et al., 2000), Eclat (Zaki, 2000), PrePost (Deng et al., 2012), và NSFI (Vo et al., 2016).

Mẫu có trọng số phổ biến (FWPs): Trong đó, cơ sở dữ liệu được quét một lần để tạo các tập hợp ID giao dịch (tidsets) theo phương pháp Eclat (Tao et al., 2003; Vo et al., 2013; Lee et al., 2017; Nguyen et al., 2016; Nguyen et al., 2022) hoặc quét hai lần để tạo cấu trúc

Cite this article as: Ngo, V. A., & Nguyen, D. H. (2025). Mining frequent patterns from dynamic weighted databases. *Ho Chi Minh City University of Education Journal of Science, 22*(10), 1887-1899. https://doi.org/10.54607/hcmue.js.22.10.5056(2025)

cây FP theo phương pháp FP-growth (Bui et al., 2018; Vo et al., 2020; Baek Y et al., 2020; Kim et al., 2021; Bui et al., 2021). Trong bài toán khai thác mẫu có trọng số phổ biến, các mặt hàng có trọng số khác nhau. Trọng số biểu thị tầm quan trọng của một mặt hàng, và các mặt hàng trong cơ sở dữ liệu giao dịch có thể có ý nghĩa khác nhau. Ví dụ, trong các ứng dụng thực tế, một số sản phẩm xuất hiện ít phổ biến hơn trong giao dịch nhưng mang lại giá trị doanh thu lớn hơn các sản phẩm khác, hoặc trong khai thác các mẫu duyệt web, các trang web có mức độ quan trọng khác nhau. Bài toán khai thác mẫu có trọng số phổ biến là tìm tất cả các mẫu có độ hỗ trợ có trọng số thỏa mãn một ngưỡng cho trước. Các hệ thống thông minh như phân tích dữ liệu thị trường và phân tích dữ liệu y sinh đều có thể tận dụng việc khai thác mẫu có trọng số phổ biến.

Các thuật toán khai thác FWUP hiện có được thực hiện trên cơ sở dữ liệu mà giá trị trọng số của mặt hàng là cố định và không thay đổi theo thời gian. Tuy nhiên, trong thực tế, tầm quan trọng của mỗi mặt hàng có thể bị ảnh hưởng bởi nhiều yếu tố khác nhau, chẳng hạn như thay đổi hành vi người tiêu dùng hoặc thay đổi chiến lược kinh doanh. Vấn đề này có thể dẫn đến việc trọng số của mỗi mặt hàng cần được điều chỉnh cho phù hợp. Ví dụ, quần áo ấm thường bán rất chạy vào đầu mùa đông, hoặc các đợt giảm giá lớn, thanh lí sản phẩm cũ để ra mắt sản phẩm mới, ngoài ra còn phụ thuộc vào các yếu tố khác. Do đó, việc đánh giá và thay đổi trọng số cho một mặt hàng cụ thể tại các thời điểm khác nhau là điều mà các ứng dụng yêu cầu. Ngoài ra, việc đặt trọng số cho các mặt hàng có thể nhằm mục đích thực hiện các chiến lược cạnh tranh trong kinh doanh tại một thời điểm, hoặc cùng một mặt hàng bán ở các nơi khác nhau có giá khác nhau. Có nhiều lí do khiến cùng một mặt hàng có trọng số thay đổi trong mỗi đơn hàng. Việc bỏ qua những đặc điểm đó làm cho các thuật toán khai thác FWP hiện tại không thể áp dụng cho các cơ sở dữ liệu thực tế, hoặc kết quả không như mong đợi.

Trong bài báo này, chúng tôi đề xuất bài toán khai thác mẫu phổ biến trên CSDL trọng số động và đề xuất thuật toán hiệu quả để giải quyết, đồng thời thực hiện các thực nghiêm trên nhiều bộ dữ liệu mẫu để chứng minh tính hiệu quả của phương pháp đề xuất.

1.2. Cơ sở lí thuyết và nghiên cứu liên quan

a. Cơ sở lí thuyết

Một bộ ba $\langle T, I, W\rangle$ được gọi là cơ sở dữ liệu định lượng. Trong đó:

- Giao dịch định lượng là một tập hợp $T=\{t_1,t_2,...,t_m\}$
- Các mặt hàng trong mỗi giao dịch là một tập hợp $I= \{i_1,i_2, ..., i_n\}$
- Và W là tập hợp các trọng số tương ứng của I, $W = \{w_1, w_2, ...,w_n\}$là tập hợp trọng số của các mặt hàng.

Bảng 1 trình bày một ví dụ về cơ sở dữ liệu định lượng. Cơ sở dữ liệu này bao gồm tổng cộng năm giao dịch t_1, t_2, t_3, t_4, t_5 được tạo từ tập hợp các mặt hàng A, B, C, D, E với trọng số tương ứng là 0.3, 0.5, 0.2, 0.4, 0.7.

Bảng 1. *Ví dụ về CSDL định lượng*

A						**B**	
CSDL giao dịch						**Trọng số mục**	
Giao dịch	***A***	***B***	***C***	***D***	***E***	**Mục**	**Trọng số**
t_1	1	0	1	0	1	*A*	0.3
t_2	0	1	1	0	1	*B*	0.5
t_3	1	0	1	0	1	*C*	0.2
t_4	1	0	1	1	0	*D*	0.4
t_5	1	1	1	1	1	*E*	0.7

Ví dụ chi tiết, giao dịch thứ 2 (t_2) = 0,1,1,0,1 có nghĩa là khách hàng đã mua một mặt hàng *B, C* và mặt hàng *E*.

Trong mục này, chúng tôi định nghĩa các khái niệm cơ bản được sử dụng trong bài báo này và mô tả bài toán khai thác các mẫu phổ biến trên CSDL định lượng động.

Một số định nghĩa về CSDL định lượng:

Định nghĩa 1. Trọng số của giao dịch t_k được tính như sau:

$$tw(t_k) = \frac{\sum_{i_j \in S(t_k)} w_j}{|S(t_k)|} \quad (1)$$

trong đó i_j và w_j lần lượt là một mục và trọng số của nó trong giao dịch t_k, còn $S(t_k)$ là tổng số mục trong giao dịch t_k. Ví dụ 1. Dựa trên Công thức (1), giá trị *tw* của giao dịch t_4 được tính như sau: $tw(t_4)=(0.4 + 0.4 + 0.2) / 3 = 0.33$

Định nghĩa 2. Đỗ hỗ trợ trọng số của tập mục (itemset) *X*:

$$ws(X) = \frac{\sum_{t_k \in t(X)} tw(t_k)}{\sum_{t_k \in T} tw(t_k)}, \quad (2)$$

Giá trị *ws* của tập mục *AD* được xác định bằng Công thức (2) như sau:

$ws(AD)= tw(t_4) + tw(t_5)] / \sum tw = (0.33 + 0.32) / 1.92 \approx 0.3385$

Dựa trên các khái niệm nền tảng này, phần tiếp theo sẽ mở rộng và định nghĩa lại các công thức cho bài toán trên cơ sở dữ liệu trọng số động, vốn là trọng tâm của nghiên cứu này

b. Mẫu phổ biến trên CSDL định lượng động

Trong một cơ sở dữ liệu trọng số động, trọng số của một mặt hàng có thể thay đổi trong bất kì lô giao dịch nào dựa trên tầm quan trọng của mặt hàng đó tại các thời điểm khác nhau. Mỗi lô giao dịch có thể bao gồm một hoặc nhiều giao dịch và tương ứng với một tập hợp các trọng số.

Do đó, một cơ sở dữ liệu trọng số động được định nghĩa như sau:

Một cơ sở dữ liệu trọng số động có dạng $\langle T, I, DW \rangle$, trong đó:

- $T = \{t_1, t_2, ..., t_m\}$ là tập hợp các giao dịch có trọng số (*m* là tổng số giao dịch).
- $I = \{i_1, i_2, ..., i_n\}$ là tập hợp các mặt hàng trong mỗi giao dịch (*n* là tổng số mặt hàng).
- $DW = \{dw_1, dw_2, ..., dw_p\}$ là tập hợp các lô trọng số (*p* là tổng số lô).

Mỗi giao dịch $t_k=\{ik_1, ik_2, ..., ik_n\}$, trong đó ik_j là mặt hàng thứ *j* trong giao dịch *tk* và tương ứng với một $dw_i \in DW$. Mỗi lô $dw_i = \{w_1, w_2, ..., w_n\}$ là một tập hợp các trọng số của các mặt hàng.

Ví dụ 2: một CSDL động dựa trên CSDL tĩnh trong Ví dụ 1 được mô tả trong Bảng 2A và Bảng 2B. Cơ sở dữ liệu này có tổng cộng năm giao dịch (t_1, t_2, t_3, t_4, t_5) và năm mặt hàng (*A*, *B*, *C*, *D*, *E*) với hai lô trọng số (dw_1, dw_2). Trọng số của các mặt hàng trong các giao dịch t_1, t_2, t_3 là dw_1={0.3, 0.5, 0.2, 0.4, 0.7}, và trọng số của các mặt hàng trong các giao dịch t_4, t_5 là dw_2={0.4, 0.3, 0.4, 0.2, 0.3}.

Bảng 2B. *Trong số của các item*

Trọng số	***A***	***B***	***C***	***D***	***E***
dw_1	0.3	0.5	0.2	0.4	0.7
dw_2	0.4	0.3	0.4	0.2	0.3

Bảng 2A. *CSDL trọng số*

CSDL giao dịch						
Giao dịch	***A***	***B***	***C***	***D***	***E***	**Trọng số**
t_1	1	0	1	1	1	dw_1
t_2	0	1	1	0	1	
t_3	1	0	1	0	1	
t_4	1	0	1	1	0	dw_2
t_5	1	1	1	1	1	

Trong cơ sở dữ liệu trọng số động, mỗi giao dịch sẽ tương ứng với một tập hợp các trọng số. Do đó, chúng ta định nghĩa lại trọng số giao dịch như sau:

Định nghĩa 3.

Trọng số giao dịch (*tw*) của một giao dịch t_k, với tập hợp trọng số d_{wp} , được định nghĩa như sau:

$$tw(t_k) = \frac{\sum_{i_j \in t} w_{pj}}{S(t_k)}, \quad (3)$$

Trong đó $w_{pj} \in d_{wp}$ là trọng số của mặt hàng i_j trong giao dịch t_k, và $S(t_k)$ là tổng số mặt hàng trong giao dịch t_k.

Ví dụ 3. Áp dụng Công thức (3), tất cả các giá trị *tw* trong cơ sở dữ liệu trọng số động (dWDB) được tính toán và thể hiện trong Bảng 3.

Bảng 3. *Tw các giao dịch của CSDL trong ví dụ 2*

Giao dịch	**Trọng số**	**Công thức**	***tw***
t_1	dw_1	(0.3 + 0.2 + 0.4 + 0.7) / 4	0.4
t_2	dw_1	(0.5 + 0.2 + 0.7) / 3	0.47
t_3	dw_1	(0.3 + 0.2 + 0.7) / 3	0.4
t_4	dw_2	(0.4 + 0.4 + 0.2) / 3	0.33
t_5	dw_2	(0.4 + 0.3 + 0.4 + 0.2 + 0.3) / 5	0.32
		Sum of *tw* values (*sumtw*)	1.92

Bài toán khai thác các mẫu phổ biến trọng số (FWP) nhằm mục đích tìm tất cả các FWP có độ hỗ trợ trọng số lớn hơn hoặc bằng một ngưỡng cho trước, được kí hiệu là *minws*. Trong đó, độ hỗ trợ có trọng số (*ws*) của mỗi mẫu được tính toán theo Công thức (3).

Để định vị rõ hơn đóng góp của bài báo, chúng tôi sẽ tổng quan một số công trình nghiên cứu tiên phong trong lĩnh vực khai thác mẫu phổ biến trên CSDL trọng số trong phần tiếp theo sau đây.

c. Một số nghiên cứu liên quan

Năm 1998, Ramkumar (Ramkumar et al., 1998) lần đầu tiên giới thiệu bài toán khai thác các mẫu có trọng số phổ biến (FWP). Sau đó, Tao và cộng sự (2003) đã đề xuất một khung khái niệm mới với các khái niệm trọng số giao dịch và độ hỗ trợ có trọng số để khai thác FWP.

Trọng số giao dịch (tw – transactions weighted) là trung bình có trọng số của các mặt hàng trong một giao dịch, còn độ hỗ trợ có trọng số là tổng các giá trị tw của các giao dịch chứa mẫu đó chia cho tổng các giá trị tw của tất cả các giao dịch. Do đó, việc khai thác FWP là tìm tất cả các mẫu có độ hỗ trợ có trọng số không nhỏ hơn một ngưỡng độ hỗ trợ có trọng số tối thiểu do người dùng chỉ định trong cơ sở dữ liệu.

Với cách tiếp cận này, Vo và cộng sự (2013) đã đề xuất cấu trúc WIT-tree như một phần mở rộng của cấu trúc IT-tree để khai thác FWP. Phương pháp này được chứng minh là hiệu quả vì nó chỉ cần quét cơ sở dữ liệu một lần để tạo các tập hợp ID giao dịch (tidset) và thực hiện khai thác trên WIT-tree. Ngoài ra, chiến lược diffset cũng được áp dụng để tăng tốc quá trình khai thác. Tuy nhiên, phương pháp này không hiệu quả trên các cơ sở dữ liệu lớn vì tốn nhiều bộ nhớ để lưu trữ tidset.

Tiếp theo, Nguyen và cộng sự (2022) sự đề xuất sử dụng cấu trúc bitset IWS (Nguyen et al., 2016) và bitset mở rộng EIWS (Nguyen et al., 2022) cùng với hướng tiếp cận Eclat để tối ưu hóa việc lưu trữ và tính giao tidset của các itemsets. Ở hướng tiếp cận FP-growth, Lee và cộng sự (Lee et al., 2017) đề xuất hai thuật toán, PWP-WSD và PWP-TCD, để khai thác FWP dựa trên cấu trúc FP-tree. Phương pháp này nén cơ sở dữ liệu thành cấu trúc cây tiền tố và không cần lưu trữ tidset. Tuy nhiên, nó phải quét cây nhiều lần để khai thác FWP.

Gần đây, Bui và cộng sự (2018) đã đề xuất cấu trúc WN-list và thuật toán NPWP để khai thác FWP. Cấu trúc WN-list mở rộng cấu trúc N-list (Deng et al., 2012) với khả năng nén dữ liệu rất hiệu quả. Kết quả thực nghiệm cho thấy thuật toán NPWP hiện là thuật toán khai thác FWP tốt nhất hiện nay (State – Of – The – Art), phương pháp này tiếp tục được mở rộng áp dụng có hiệu quả cho các bài toán khai thác top-rank-k (Vo et al., 2020), khai thác tập đóng (Bui et al., 2020) hay CSDL dạng chuỗi (Bui et al., 2021).

Tuy nhiên, các nghiên cứu trên chỉ tập trung vào các CSDL trọng số tĩnh, có nghĩa là trọng số trong các giao dịch không có sự thay đổi. Trong bài báo này chúng tôi đề xuất bài toán khai thác mẫu trên CSDL có trọng số thay đổi (CSDL định lượng động).

2. Nội dung nghiên cứu

Trong phần này chúng tôi sẽ trình bày về thuật toán khai thác mẫu phổ biến cho CSDL trọng số động - dFWNL, và một ví dụ cụ thể cho thuật toán này.

2.1. Thuật toán khai thác mẫu phổ biến trên CSDL trọng số động

Trong phần này chúng tôi đề xuất Thuật toán **dFWNL** (dynamic Frequent Weighted pattern mining using weighted utility N-list structure) sử dụng cấu trúc dữ liệu WN-tree, WNList (Bui et al., 2018 & Bui et al., 2021) để nén dữ liệu và khai thác hiệu quả mẫu phổ

biến trọng số. Thuật toán này kết hợp phương pháp duyệt cây liệt kê tập hợp (set-enumerations-tree) và các kĩ thuật tối ưu hóa để giảm không gian tìm kiếm.

Các bước thực hiện

Bước 1. Xây dựng cây WN-Tree

- Đầu tiên, thuật toán quét CSDL để tìm tập các 1-FWP (kí hiệu là I1) và sắp xếp chúng theo thứ tự ws giảm dần.
- Các mục trong mỗi giao dịch được sắp xếp lại theo thứ tự của I1, và các mục không thuộc I1 sẽ bị loại bỏ.
- Một cây WN-tree được xây dựng bằng cách chèn các giao dịch đã được sắp xếp vào cây.
- Cuối cùng, thuật toán duyệt cây để gán chỉ số pre-order (pre) và post-order (post) cho mỗi nút trong cây.

Bước 2. Tạo WNList cho các 1-pattern

- Từ cây WN-tree hoàn chỉnh, thuật toán tạo ra một WNList cho mỗi mục trong I1.
- WNList của một mục là một chuỗi các WN-code (bộ ba pre, post, weight) của tất cả các nút đại diện cho mục đó trên cây.

Bước 3. Tính toán hỗ trợ cho các 2-pattern

- Để tối ưu hóa, thuật toán sử dụng một mảng hai chiều F2 để lưu trữ trước trọng số của các 2-pattern. Giá trị này được tính bằng cách duyệt cây WN-tree và kiểm tra mối quan hệ tổ tiên-hậu duệ giữa các nút.

Bước 4. Khai thác đệ quy (Thủ tục Find_FWP)

- Thuật toán duyệt qua các mục trong I1 để tạo ra các FWP dài hơn. Với mỗi tiền tố, nó sử dụng hai danh sách: *L* (cho các mẫu cần tính toán giao WNList) và *S* (cho các mẫu có thể suy ra nhanh).
- **Chiến lược tối ưu (Find_FWP_sameWS)**: Dựa trên ki thuật: nếu ws($P \cup \{i\}$) == *ws*(*P*), có thể suy ra rằng mọi giao dịch chứa *P* cũng chứa *i*. Do đó, các FWP mở rộng từ *P* với các mục trong tập *S* có thể được tạo ra trực tiếp mà không cần thực hiện phép giao WNList tốn kém.
- Thủ tục Find_FWP được gọi đệ quy với các tiền tố mới cùng hai danh sách *L* và *S* tương ứng để tiếp tục tìm kiếm.

Các bước trên được thể hiện qua giả mã trong Hình 1 như sau:

dFWNL Algorithm

Input: A dynamic weighted database $dWDB$ and a threshold $minws$
Output: $FWPs$, the set of all frequent weighted patterns

1. Call **Build_WN_Tree**(dWDB, *minws*) to create I_1 and WN-Tree;
2. $FWPs \leftarrow I_1$
3. Scan Tree to create WNList of the item in I_1
4. Create $F2 =$ double $[|I_1|][|I_1|]$
5. Scan WN-Tree by the pre-order ranking do
6. | **for** each node *N* registering item i_x **do**
7. | | **for** each ancestor N_a of *N* registering item i_y **do**

$F2[i_x][i_y]$ += $N.weight$
for $x = |I_1| - 1$ downto 1 **do**
  $Prefix_2 \leftarrow \{i_x\}$, $L_2 \leftarrow \emptyset$, $S_2 \leftarrow \emptyset$
  for $y = 0$ to $x - 1$ **do**
    $ws(i_{xy}) \leftarrow F2[i_x][i_y]/sumtw$
    if $ws(i_{xy}) == ws(i_x)$ **then** $S_2 \leftarrow i_y$
    else if $ws(i_{xy}) \geq minws$ **then**
      $i_{xy} \leftarrow i_y$
      $wl(i_{xy}) \leftarrow$ **WNList_intersection**$(wl(i_x), wl(i_y))$
      $L_2 \leftarrow i_{xy}$
    $FWPs \leftarrow i_x i_y$
  if $|S_2| > 0$ **then Find_FWP_sameWS**$(Prefix_2, S_2)$
  if $|L_2| > 0$ **then Find_FWP**$(Prefix_2, L_2, S_2)$
return $FWPs$

Procedure Find_FWP $(Prefix_k, L_k, S_k)$
$Prefix_{next} \leftarrow Prefix_k$
for $i = |L_k| - 1$ downto 1 **do**
  $Prefix_{next} \leftarrow L_k[i]$, $L_{next} \leftarrow \emptyset$, $S_{next} \leftarrow S_k$
  for $j = 0$ to $i - 1$ **do**
    $C \leftarrow L_k[j]$
    $wl(C) \leftarrow$ **WNList_intersection**$(wl(L_k[i]), wl(L_k[j]))$
    if $wl(C) \neq NULL$ **then**
      if $ws(C) == ws(L_k[i])$ **then** $S_{next} \leftarrow L_k[j]$
      else
        $L_{next} \leftarrow C$
        $FWPs \leftarrow \{Prefix_k \cup L_k[i] \cup L_k[j]\}$
  if $|S_{next}| > 0$ **then Find_FWP_sameWS**$(Prefix_{next}, S_{next})$
  if $|L_{next}| > 0$ **then Find_FWP**$(Prefix_{next}, L_{next}, S_{next})$
  remove last item of $Prefix_{next}$
$Prefix_{next} \leftarrow L_k[0]$
if $|S_k| > 0$ **then Find_FWP_sameWS**$(Prefix_{next}, S_k)$

Procedure Find_FWP_sameWS $(Prefix, S)$
Generate $Child$ which is the set of all subsets of S
for each $s_i \in Child$ do
  $FWPs \leftarrow \{Prefix \cup s_i\}$

Function WNList_intersection(wl_1, wl_2)
$wl_3 \leftarrow \emptyset, SC \leftarrow 0$;
$i \leftarrow 0, j \leftarrow 0$;
$CR_1 = ws(PX)$; $CR_2 = ws(PY)$;
while $i < |wl_1|$ *and* $j < |wl_2|$ **do**
  if $wl_2[j].pre < wl_1[i].pre$ **then**
    if $wl_2[j].post > wl_1[i].post$ **then**
      if $|wl_3| > 0$ *and* $wl_3[|wl_3| - 1].pre = wl_2[j].pre$ **then**
        $wl_3[|wl_3| - 1].weight$+= $wl_1[i].weight$;
      else
        Add $\{wl_2[j].pre, wl_2[j].post, wl_1[i].weight\}$ into wl_3;
      SC+= $wl_1[i].weight$; CR_1−= $wl_1[i].weight$; i++;
    else CR_2−= $wl_2[j].weight$; j++;

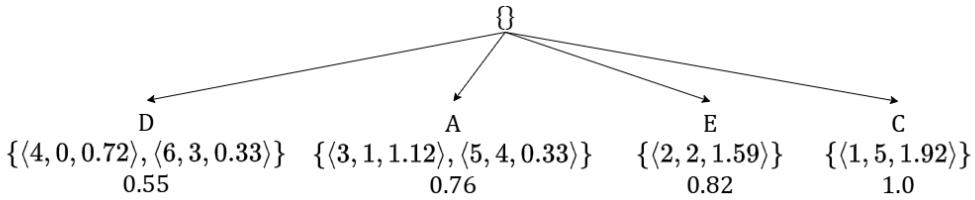
15. **else** $CR_1 -= wl_1[i].weight$; i++;
16. **if** $SC + \min(CR_1, CR_2) < (minws \times sumtw)$ **then return** $NULL$;
17. **return** wl_3;

Hình 1. *Giả mã thuật toán khai thác mẫu phổ biến trên CSDL trọng số động*

2.2. Một ví dụ cụ thể

Trong phần này, chúng tôi sẽ trình bày một ví dụ cụ thể đối với thuật toán đề xuất trong phần 2.1.

Với dWDB trong Bảng 2 và minws = 0.45, thuật toán dFWNL đầu tiên sử dụng thủ tục Build_WN_Tree để xây dựng cây WN; các 1-FWPs I_1={*C*, *E*, *A*, *D*} và tổng trọng số (sumtw) = 1.92. I_1 được thêm vào kết quả FWUP: FWPs={C, E, A, D}. Tiếp theo, cây WN và I_1 được sử dụng để tạo danh sách WN của I_1 (Hình 2), và F2 lưu trữ trọng số của các mẫu 2-itemset.

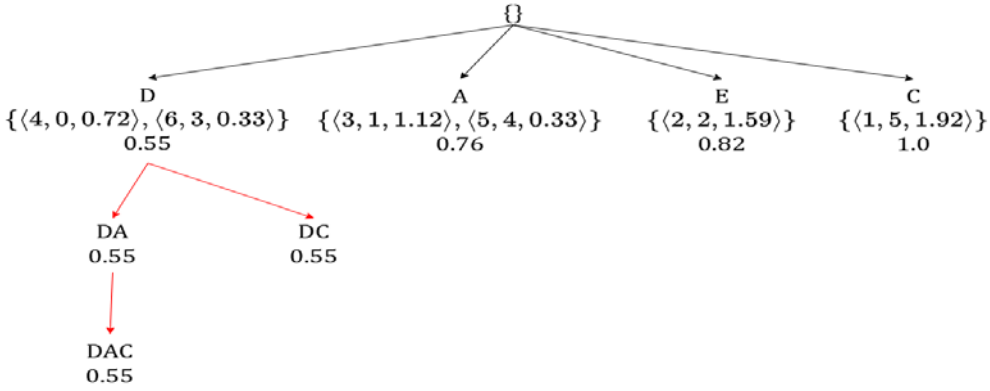


Hình 2. *I_1 và WNlist tương ứng với minws=0.45*

Bước 1. Xem xét nhánh *D*: Prefix_2 = {*D*}, S_2 = ∅, L_2 = ∅, chúng ta có:

- *ws*(*DA*) = F2(*DA*)/1.92 = 1.05/1.92 = *ws*(*D*) nên *A* được thêm vào S_2.
- *ws*(*DE*) = F2(*DE*)/1.92 = 0.38/1.92 < 0.5 (không thỏa mãn).
- *ws*(*DC*) = F2(*DC*)/3.21 = 1.05/1.92 = *ws*(*D*) nên *C* được thêm vào S_2.

Vì S_2 = {*A*, *C*} ≠ ∅, chúng ta gọi thủ tục Find_FW P_sameWS(Prefix_2={*D*}, S_2 = {*A*, *C*}) để cập nhật FWPs: FWPs ={*C*, *E*, *A*, *D*, *DA*, *DC*, *DAC*}. Kết quả của bước này được trình bày trong Hình 3. Các nút có mũi tên màu đỏ được tạo bởi Find_FWP_sameWS mà không cần giao WNList.



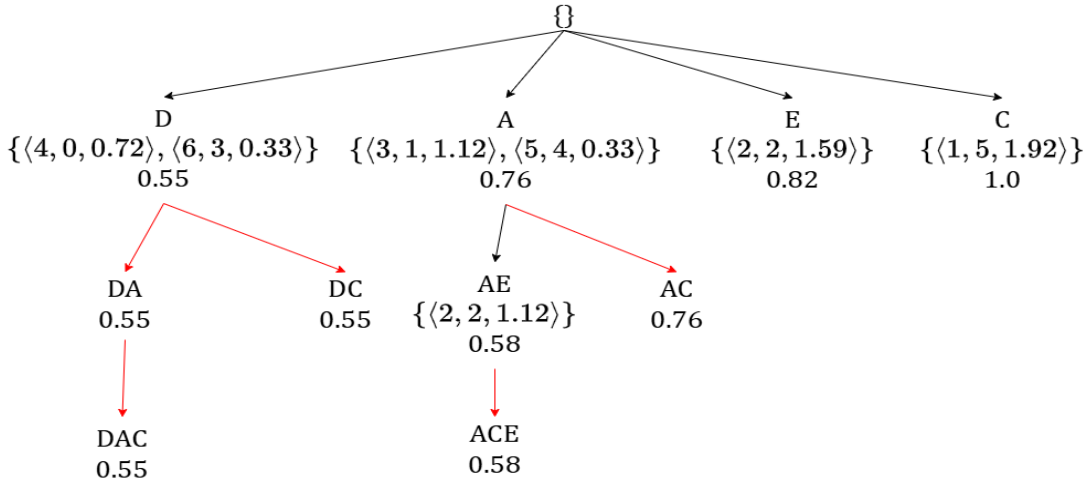
Hình 3. *Kết quả của sự kết hợp giữa D và các 1-FWP còn lại*

Bước 2. Xem xét nhánh *A*: Prefix_2 = {*A*}, S_2=∅, L_2 = ∅, chúng ta có:

- *ws*(*AE*) = F2(*AE*)/1.92 = 1.12/1.92 > 0.4 và ws(*AE*) ≠ *ws*(*A*) nên *E* được thêm vào L_2, wl(*AE*)={(2,2,1.12)}, và *AE* được thêm vào FWPs.
- *ws*(*AC*) = F2(*AC*)/1.92 = 1.45/1.92 = *ws*(*A*) nên *C* được thêm vào S_2.

Vì S_2 = {C} ≠ ∅, chúng ta gọi thủ tục Find_FWP_sameWS(Prefix_2 = {*A*},S_2 = {*C*}) để cập nhật FWPs: FWPs ={*C, E, A, D, DA, DC, DAC, AE, AC*}.

Mặt khác, L_2 = {*E*} ≠ ∅, vì vậy chúng ta gọi thủ tục Find_FWP(Prefix_2 = {*A*}, L_2 = {*E*}, S_2 = {*C*}). Vì L_2 chỉ có một phần tử, thủ tục Find_FWP_sameWS({*AE*}, {*C*}) được gọi để cập nhật FWPs: FWPs = {*C, E, A, D, DA, DC, DAC, AE, AC, AEC*} (Hình 4).



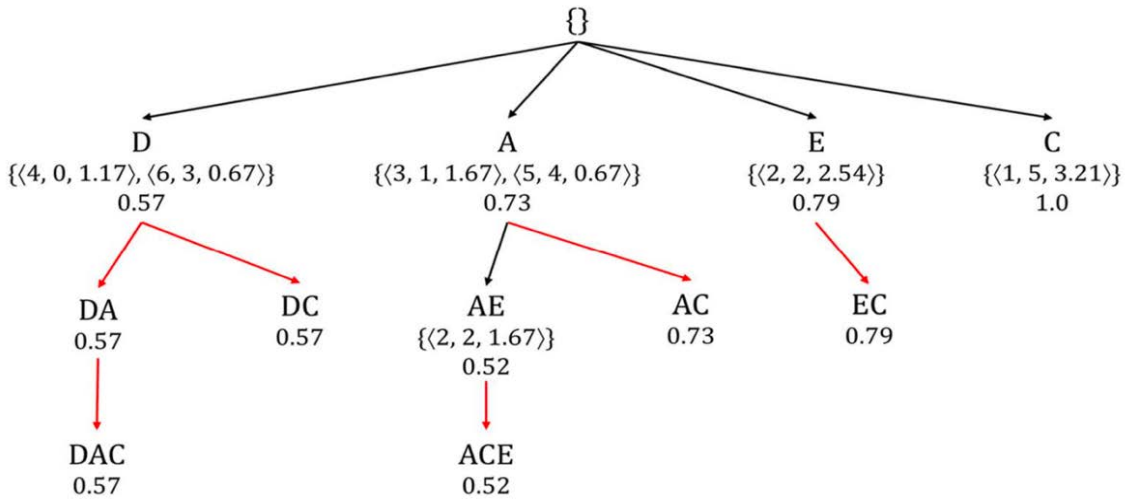
Hình 4. *Kết quả của sự kết hợp giữa A với các 1-FWP còn lại*

Bước 3. Xem xét nhánh E: Prefix_2={*E*}, S_2=∅, L_2=∅, chúng ta có:

- *ws*(*EC*) = F2(*EC*)/1.92 = 1.59/1.92 = *ws*(*E*) nên C được thêm vào S_2.

Vì S_2 = {*C*} ≠ ∅, chúng ta gọi thủ tục Find_FWP_sameWS(Prefix_2={*E*}, S_2 = {*C*}) để cập nhật FWPs: FWPs ={*C, E, A, D, DA, DC, DAC, AE, AC, AEC, EC*}.

Thuật toán kết thúc, kết quả cuối cùng là 11 FWP như sau: FWPs={*C, E, A, D, DA, DC, DAC, AE, AC, AEC, EC*}. Cây liệt kê tập hợp của quá trình khai thác được thể hiện trong Hình 5.



Hình 5. *Tất cả các FWP của CSDL trong Bảng 2 với minws=0.45*

3. Kết quả và thảo luận

3.1. Cơ sở thực nghiệm

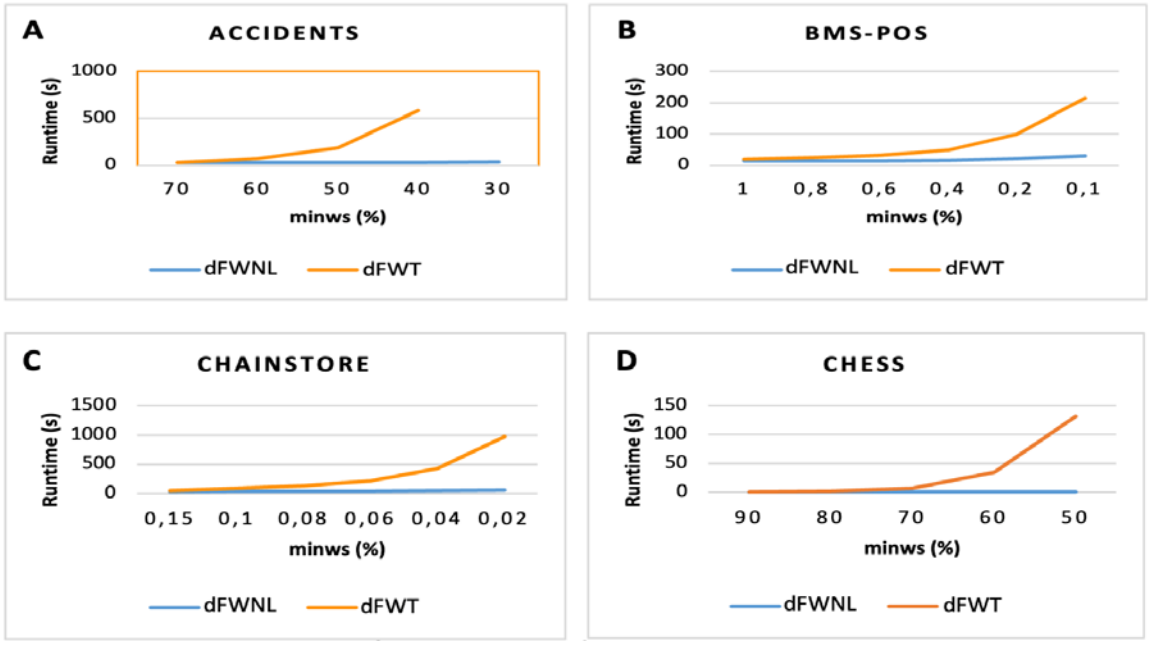
Các thử nghiệm được thực hiện trên một máy tính có bộ xử lí Intel Core i7-7700HQ CPU @ 2.80GHz, RAM 16 GB. Các thuật toán được cài đặt bằng Java Development Kit (JDK) 8. Chúng tôi sử dụng một số cơ sở dữ liệu tổng hợp và thực tế phổ biến trong lĩnh vực khai thác dữ liệu được trình bày trong Bảng 4.

Bảng 4. *Dữ liệu thực nghiệm*

CSDL	Số mục	Số giao dịch	Số giao dịch mỗi lô
Accidents	468	340,183	5000
BMS-POS	1657	515597	50000
Chainstore	46,086	1,112,949	80000
Chess	41,270	990,002	100000

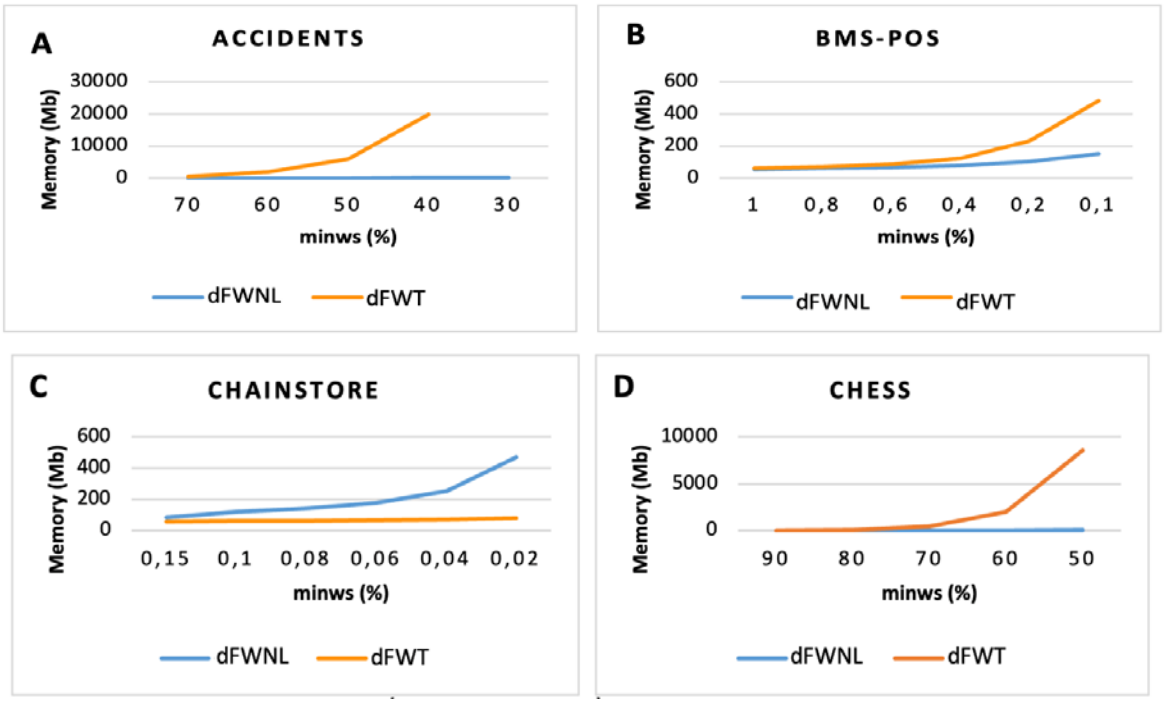
Trong phần thực nghiệm này, chúng tôi so sánh thuật toán đề xuất trên với dFWT - một thuật toán được thiết kế dựa trên tiếp cận Eclat (Zaki, 2000) với cấu trúc bitsets mở rộng EIWS (Nguyen et al., 2022), đây là phương pháp có hiệu quả nhất theo tiếp cận Eclat, với các kĩ thuật cắt bỏ các bít 0 và tính nhanh giao nhanh các tidset. Các kết quả thực nghiệm trên 04 dữ liệu mô tả trong Bảng 4 được trình bày trong phần tiếp theo.

3.2. So sánh thời gian thực hiện của hai thuật toán



Hình 6. *Kết qủa thực nghiệm về thời gian trên các bộ dữ liệu*

3.3. So sánh bộ nhớ của hai thuật toán



Hình 7. *Kết qủa thực nghiệm về bộ nhớ trên các bộ dữ liệu*

3.4. Thảo luận

Kết quả thực nghiệm thể hiện trên Hình 6 và Hình 7 cho thấy thuật toán dFWNL hiệu quả hơn hẳn về mặt thời gian. Thuật toán dFWN sử dụng cấu trúc WN-list (Bui et al., 2018) được phát triển từ cấu trúc N-list (Deng et al., 2012), đây là cấu trúc hiệu quả nhất cho đến hiện nay trong khai thác mẫu. Thuật toán dFWT dựa trên tiếp cận Eclat (Zaki, 2000) có sử dụng cấu trúc bảng bít mở rộng EIWS (Nguyen et al., 2022) khá hiệu quả nhưng cách tiếp cận này có hạn chế lớn với các CSDL lớn có nhiều giao dịch sẽ phát sinh bảng bít có độ dài lớn.

Về bộ nhớ sử dụng trong quá trình khai thác, thuật toán dFWNL không hiệu quả bằng dFWT do đây là CSDL lớn (hơn một triệu giao dịch) nên quá trình nén CSDL lên cây tốn nhiều bộ nhớ. Mặc dù việc xây dựng cây ban đầu đòi hỏi nhiều bộ nhớ hơn, cấu trúc WN-Tree nén của dFWNL lại mang lại lợi thế tốc độ vượt trội trong giai đoạn khai thác, hiệu quả hơn đáng kể so với phương pháp duyệt tidset của dFWT

4. Kết luận

Trong bài báo này, chúng tôi đã giới thiệu một vấn đề mới liên quan đến việc khai thác các mẫu phổ biến trọng số trên cơ sở dữ liệu trọng số (dWDBs). Để giải quyết vấn đề này, chúng tôi đề xuất thuật toán mới dFWNL. Thuật toán dFWT sử dụng cấu trúc dữ liệu tidset để xử lí các mục có trọng số động, trong khi dFWNL sử dụng cấu trúc dữ liệu WNList hiệu quả hơn để tối ưu hóa quá trình khai thác. Các thử nghiệm toàn diện đã chứng minh rằng dFWNL vượt trội hơn dFWT về thời gian thực thi, khẳng định tính hiệu quả của hướng tiếp cận dựa trên cấu trúc WN-List cho bài toán khai thác trên CSDL trọng số động.

Thời gian tới chúng tôi sẽ tiếp tục nghiên cứu áp dụng các phương pháp song song hoá để đạt được sự tối ưu tốt hơn cho bài toán này, đồng thời nghiên cứu áp dụng cho các CSDL động lớn và phức tạp hơn.

- ***Tuyên bố về quyền lợi:*** *Các tác giả xác nhận hoàn toàn không có xung đột về quyền lợi.*
- ***Lời cảm ơn:*** *Nghiên cứu này được tài trợ bởi Trường Đại học Sài Gòn, mã số đề tài CSB2024-01.*

TÀI LIỆU THAM KHẢO

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, *22*(2), 207-216. https://doi.org/10.1145/170036.170072

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, 487-499.

Baek Y, Yun U, Lin JCW, Yoon E, Fujita H (2020) Efficiently mining erasable stream patterns for intelligent systems over uncertain data. *Int J Intell Syst*, 35(11), 1699-1734. https://doi.org/10.1002/int.22269

Bui, H., Vo, B., Nguyen, H., Nguyen-Hoang, T. A., & Hong, T. P. (2018). A weighted N-list-based method for mining frequent weighted itemsets. *Expert Systems with Applications*, *96*, 388-405. https://doi.org/10.1016/j.eswa.2017.10.039

Deng, Z., Wang, Z., & Jiang, J. (2012). A new algorithm for fast mining frequent itemsets using N-lists. *Science China Information Sciences*, *55(9)*, 2008-2030. https://doi.org/10.1007/s11432-012-4638-z

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 1-12. https://doi.org/10.1145/335191.33537

Kim, H., Yun, U., Baek, Y., Kim, H., Nam, H., Lin, J. C., & Fournier-Viger, P. (2021) Damped sliding based utility oriented pattern mining over stream data. *Knowl-Based Syst*. https://doi.org/10.1016/j.knosys.2020.106653Get rights and content

Lee, G., Yun, U., & Ryu, K. (2017). Mining frequent weighted itemsets without storing transaction IDs and generating candidates. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *25(1)*, 111-144. https://doi.org/10.1142/S0218488517500052

Nguyen, H., Le, T., Nguyen, M., Fournier-Viger, P., Tseng, V. S., & Vo, B. (2022). Mining frequent weighted utility itemsets in hierarchical quantitative databases. *Knowledge-Based Systems*, *237*, 107709. https://doi.org/10.1016/j.knosys.2021.107709

Nguyen, H., Vo, B., Nguyen, M., & Pedrycz, W. (2016). An efficient algorithm for mining frequent weighted itemsets using interval word segments. *Applied Intelligence*, *45(4)*, 1008-1020. https://doi.org/10.1007/s10489-016-0799-6

Tao, F., Murtagh, F., & Farid, M. (2003). Weighted association rule mining using weighted support and significance framework. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 661-666. https://doi.org/10.1145/956750.956836

Vo, B., Bui, H., Vo, T., & Le, T. (2020). Mining top-rank-k frequent weighted itemsets using WN-list structures and an early pruning strategy. *Knowledge-Based Systems*, 201-202, 106064. https://doi.org/10.1016/j.knosys.2020.106064

Vo, B., Coenen, F., & Le, B. (2013). A new method for mining frequent weighted itemsets based on WIT-Tree. *Expert Systems with Applications, 40(4),* 1256-1264. https://doi.org/10.1016/j.eswa.2012.08.065

Vo, B., Le, T., Coenen, F., & Hong, T. (2016). Mining frequent itemsets using the N-list and subsume concepts. *International Journal of Machine Learning and Cybernetics*, *7(2)*, 253-265. https://doi.org/10.1007/s13042-014-0252-2.

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering, 12(3),* 372-390. https://doi.org/10.1109/69.846291

MINING FREQUENT PATTERNS FROM DYNAMIC WEIGHTED DATABASES

Ngo Viet Anh, Nguyen Duy Ham*

Saigon University, Vietnam

**Corresponding author: Nguyen Duy Ham – Email: ndham@sgu.edu.vn*

Received: June 11, 2025; Revised: September 8, 2025; Accepted: September 16, 2025

ABSTRACT

Pattern mining is one of the fundamental tasks in modern data mining. In particular, mining frequent weighted patterns (PWPs) from quantitative databases is an important problem aimed at discovering frequent patterns based on item weights. However, existing studies have largely overlooked the scenario in which item weights vary over time — known as dynamic weighted databases (dWDBs). In many real-world applications, item weights fluctuate to reflect shifts in their significance, such as profit margins or temporal importance (e.g., air conditioners are more popular in summer, and medical masks become critical during respiratory disease outbreaks).

In this paper, we introduce a new problem of mining PWPs from quantitative databases with dynamically changing item weights — referred to as dynamic quantitative databases. To address this problem, we first propose an algorithm named dPWPT, built upon the traditional tidset structure. We then present a second algorithm, dFWNL, which utilizes a novel data structure called dWNList to efficiently mine PWPs from dWDBs. Finally, we conduct extensive experiments on various dWDBs to evaluate and validate the effectiveness of our proposed algorithms.

Keywords: Dynamic weighted databases; frequent patterns; WNList structure; Weighted databases