# MỘT THUẬT TOÁN HIỆU QUẢ ĐỂ TRÍCH XUẤT TẬP SKYLINE

**Nguyễn Thị Thanh Thủy, Mạnh Thiên Lý, Nguyễn Văn Lễ và Vũ Văn Vinh**
*Trường Đại học Công nghiệp Thực phẩm TP. Hồ Chí Minh*
Ngày nhận bài 06/10/2020, ngày nhận đăng 23/12/2020

**Tóm tắt**: Khai thác tập hữu ích phổ biến thuộc đường chân trời (Skyline frequent-utility patterns (SFUPs)) là việc khám phá các tập mặt hàng (itemset) vượt trội hơn các tập mặt hàng khác về cả tần số và độ hữu ích trong cơ sở dữ liệu giao dịch. Trong những năm gần đây, nhiều thuật toán đã được đề xuất nhằm khai thác tập hữu ích phổ biến thuộc đường chân trời, trong đó *SkyFUP* là thuật toán hiệu quả nhất. Tuy nhiên, thuật toán *SkyFUP* vẫn còn những hạn chế cả về thời gian thực thi và không gian lưu trữ. Trong bài báo này, nhóm tác giả đề xuất thuật toán *SkyMiner* để khai thác tập SFUPs hiệu quả hơn bằng cách sử dụng cấu trúc lưu trữ *utility-list* kết hợp với các chiến lược cắt tỉa nhằm làm giảm đáng kể số lượng các ứng viên cần phải tìm kiếm trong quá trình khai thác. Kết quả thực nghiệm cho thấy thuật toán *SkyMiner* có hiệu suất thực thi tốt hơn thuật toán mới nhất là *SkyFUP* về thời gian thực thi, bộ nhớ sử dụng và số lượng các ứng viên được tạo ra.

**Từ khóa**: SFUPs; tập hữu ích phổ biến thuộc đường chân trời; EUCS; LA.

## 1. Giới thiệu

Trong bối cảnh kinh tế xã hội ngày càng phát triển như hiện nay thì nhu cầu mua sắm của khách hàng ngày càng đa dạng và phong phú. Sự cạnh tranh giữa các doanh nghiệp trong việc thu hút khách hàng và tối đa hóa lợi nhuận ngày càng khốc liệt. Để giúp các doanh nghiệp khai thác thông tin hữu ích từ thói quen mua hàng của khách hàng, các nghiên cứu về khai thác tập phổ biến (Frequent Pattern Mining - FIM) [1-4] và khai thác luật kết hợp (Association Rule Mining - ARM) [5-7] đã được thực hiện để tìm ra tập các mặt hàng thường được khách hàng mua cùng nhau trong cơ sở dữ liệu giao dịch. Tuy nhiên, các nghiên cứu này chỉ quan tâm đến các tập mặt hàng dựa vào tần suất xuất hiện của các tập mặt hàng đó trong cơ sở dữ liệu giao dịch mà không xem xét đến lãi suất, lợi nhuận, trọng lượng cũng như những rủi ro của chúng. Do đó, các nghiên cứu về khai thác tập hữu ích cao (Mining High Utility Itemsets - HUIM) ra đời nhằm khai thác và tìm kiếm các tập mặt hàng mang lại lợi nhuận cao cho các nhà bán lẻ bằng cách quan tâm cả số lượng và lợi nhuận của các mặt hàng [8-12]. Bên cạnh đó, để tối ưu hóa lợi nhuận và định hướng các chiến lược kinh doanh, ban đầu các nghiên cứu về top-k dựa trên luật kết hợp (Top-k Association-Rule Mining) [13-15] được thực hiện để giúp các nhà quản lý tìm ra tập các mặt hàng được khách hàng mua nhiều lần nhất. Sau đó, các nghiên cứu về top-k trên tập hữu ích cao (Top-k High-Utility) [16-18] cũng được thực hiện để tìm ra các tập mặt hàng mang lại lợi nhuận cao nhất cho doanh nghiệp. Mặc dù các nghiên cứu về top-k rất hữu ích trong thực tế, nhưng các nghiên cứu này chỉ quan tâm đến một trong hai khía cạnh là tần suất hoặc độ hữu ích mà chưa có sự kết hợp giữa hai yếu tố này. Để giải quyết vấn đề này, một số thuật toán gần đây đã được đề xuất nhằm khai thác các tập mặt hàng vượt trội hơn tất cả các tập mặt hàng khác cả về tần suất

Email: thuyntt@hufi.edu.vn (N. T. T. Thủy)

và độ hữu ích, gọi là tập hữu ích phổ biến thuộc đường chân trời (SFUPs) như SKYMINE [19], SFU-Miner [20], SKYFUP-D [21]. Tuy nhiên, các thuật toán này vẫn tốn kém về cả thời gian thực hiện và không gian bộ nhớ. Trong bài báo này, chúng tôi đề xuất một thuật toán mới có tên là *SkyMiner* để tìm ra tập SFUPs hiệu quả hơn các thuật toán trước đây bằng cách áp dụng các chiến lược cắt tỉa nhằm giảm thời gian thực hiện và không gian lưu trữ trong quá trình thực thi thuật toán.

Những đóng góp chính và quan trọng của bài báo này bao gồm:

- Sử dụng cấu trúc *utility-list* để lưu trữ thông tin về độ hữu ích và tần suất xuất hiện của các tập mặt hàng, làm cơ sở cắt tỉa trong quá trình khai thác.
- Đề xuất thuật toán *SkyMiner* để khai thác tập *SFUPs* một cách hiệu quả. Áp dụng các chiến lược cắt tỉa *U-Prune, LA-Prune và EUCS-Prune* trong quá trình khai thác giúp giảm thời gian tìm kiếm và không gian lưu trữ.
- Kết quả thử nghiệm trên các bộ dữ liệu thưa, dày và bộ dữ liệu tăng trưởng theo độ lớn đã chứng tỏ rằng thuật toán *SkyMiner* mà chúng tôi đề xuất có hiệu suất thực hiện tốt hơn thuật toán mới nhất *SKYFUP* về thời gian thực thi, bộ nhớ sử dụng và số lượng ứng viên phát sinh.

Bài báo này được cấu trúc như sau: Phần 2 trình bày các nghiên cứu liên quan đến khai thác tập SFUPs. Phần 3 trình bày các định nghĩa quan trọng trong khai thác tập SFUPs. Phần 4 trình bày thuật toán *SkyMiner*. Phần 5 trình bày kết quả thực nghiệm của thuật toán. Cuối cùng, kết luận và hướng nghiên cứu trong tương lai được trình bày trong phần 6.

## 2. Các công trình liên quan

Trong phần này, chúng tôi trình bày các công trình nghiên cứu liên quan đến các vấn đề sẽ được đề xuất trong bài báo này, bao gồm: tập phổ biến (Frequent Itemsets - FIs), tập hữu ích cao (High Utility Itemsets - HUIs) và SFUPs.

### *2.1. Tập phổ biến*

Từ những năm 90 của thế kỷ XX, nhiều nghiên cứu đã tập trung vào lĩnh vực khai thác tập phổ biến. Khai thác tập phổ biến là tìm ra những tập hợp chứa những mục, tập mục xuất hiện thường xuyên cùng nhau. Năm 1994, Agrawal và cộng sự đã đề xuất thuật toán Apriori [5] để khám phá tất cả các luật kết hợp quan trọng giữa các mặt hàng trong một cơ sở dữ liệu giao dịch lớn. Apriori là thuật toán phổ biến trong các phương pháp tiếp cận theo cấp độ (level-wise approach) với các ứng viên được tạo ra ở nhiều mức. Nhiều nghiên cứu khác về tập phổ biến cũng đã được thực hiện [1-4]. Tuy nhiên, các nghiên cứu về tập phổ biến trong FIM và ARM chỉ chú trọng đến tần suất xuất hiện của các tập mặt hàng mà ít quan tâm đến các yếu tố khác như: lợi nhuận (unit profit), trọng lượng (weight) hay độ thú vị (interestingness) của các mặt hàng.

### *2.2. Tập hữu ích cao*

HUIM là một hướng nghiên cứu mở rộng của FIM khi xem xét cả số lượng (quanlity) và lợi nhuận (profit) của các mặt hàng trong cơ sở dữ liệu để khai thác các tập mục hữu ích cao. Khai thác HUIs là khám phá tập hợp chứa tất cả các tập mặt hàng thỏa mãn một ngưỡng độ hữu ích tối thiểu cho trước, ký hiệu là *minUtil*. Thông thường, ngưỡng độ hữu ích tối thiểu này do người dùng xác định. Trong thời gian qua, nhiều nghiên cứu đã được thực hiện trong lĩnh vực này và đã có nhiều thuật toán được đề xuất để nâng cao hiệu quả trong vấn đề khai thác HUIs.

Trong giai đoạn đầu, khai thác HUIs chủ yếu được thực hiện trên hai pha, pha 1 thực hiện tìm các ứng viên có TWU cao, pha 2 khai thác HUIs từ danh sách các ứng viên tìm được. Các thuật toán 2 pha điển hình như: Two-Phase [8], TWU-Mining [22], UP-Growth [10], UP-Growth+ [23]. Việc khai thác HUI bởi các thuật toán hai pha mất nhiều thời gian và lãng phí bộ nhớ khi số lượng ứng viên được tìm thấy có thể rất lớn nhưng số lượng HUI nhận được nhỏ. Để giải quyết vấn đề này, năm 2012, Liu và cộng sự đã đề xuất một thuật toán để khai thác HUI có tên là HUI-Miner (High Utility Itemset Miner) [24]. Ngoài ra, một cấu trúc lưu trữ là utility-list đã được đề xuất để lưu trữ cả thông tin hữu ích của một mặt hàng và thông tin heuristic nhằm cắt tỉa không gian tìm kiếm. Mỗi bộ trong utility-list gồm 3 thành phần: mã giao dịch (tid) chứa tập mặt hàng, độ hữu ích của tập mặt hàng trong một giao dịch (iutil) và giá trị hữu ích còn lại trong giao dịch (rutil). Năm 2017, Zida và cộng sự đề xuất thuật toán EFIM [25] khai thác tập hữu ích cao hiệu quả với việc đề xuất hai kỹ thuật nhằm giảm thời gian khai thác HUI là phép chiếu giao dịch (High-utility Database Projection - HDP) và phép trộn giao dịch (High-utility Transaction Merging - HTM). Ngoài ra, nhóm tác giả EFIM cũng đề xuất hai ngưỡng giới hạn trên (upper-bounds) nhằm thu gọn không gian tìm kiếm, đó là: cây con độ hữu ích (sub-tree utility) và độ hữu ích cục bộ (local utility). Cũng trong năm 2017, thuật toán HMiner được đề xuất bởi Krishnamoorthy [26], trong đó nổi bật là cấu trúc dữ liệu Compact Utility List (CUL) kết hợp với nhiều chiến lược cắt tỉa khác nhau để khai thác HUI một cách hiệu quả.

### *2.3. Tập hữu ích phổ biến thuộc đường chân trời*

Các thuật toán về FIM và HUIM đã được đề xuất đều cho thấy hiệu quả khai thác cao. Nhiều nghiên cứu cũng được thực hiện nhằm xem xét kết hợp cả tần suất xuất hiện và độ hữu ích của các tập mặt hàng cùng nhau trong khai thác dữ liệu [27, 28]. Tuy nhiên, các thuật toán này chủ yếu xác định tập các mục hữu ích phổ biến dựa vào hai ngưỡng là độ hữu ích tối thiểu và ngưỡng hỗ trợ tối thiểu. Việc xác định các ngưỡng chính xác trong bài toán khai thác dữ liệu luôn là một vấn đề khó khăn và quan trọng. Một hướng nghiên cứu mới được thực hiện nhằm khai thác các tập mặt hàng hữu ích theo quan điểm ưu tiên người sử dụng, theo đó tích hợp ý tưởng về các truy vấn đường chân trời trong việc khai thác mẫu. Đường chân trời là một tập hợp các điểm mà mỗi điểm không bị thống trị (dominate) bởi các điểm khác dựa trên nhiều chiều. Nhiều nghiên cứu đã được thực hiện để khai thác mẫu sử dụng khái niệm đường chân trời. Năm 2015, Goyal và cộng sự đề xuất thuật toán SKYMINE để tìm các tập hữu ích phổ biến thuộc đường chân trời (SFUP) [19]. SFUP là những tập mặt hàng không bị thống trị bởi các tập mặt hàng khác. Việc tìm ra SFUPs được thực hiện bằng cách xem xét độ hữu ích và tần suất xuất hiện của các tập mặt hàng. Thuật toán SKYMINE dựa trên cấu trúc cây UP-Tree và gồm hai giai đoạn được gọi là Filter và Refine. Năm 2017, Pan và cộng sự đề xuất thuật toán SFU-Miner khai thác SFUPs trên hai pha [20]. Pha đầu tìm tất cả các ứng viên dự kiến là SFUP, pha 2 duyệt qua tất cả các ứng viên để xác định ứng viên nào là SFUP. Năm 2018, hai thuật toán SKYFUP-D và SKYFUP-B được đề xuất bởi Lin và cộng sự [21] để khai thác SFUPs. Ngoài ra, một cấu trúc utility-list hiệu quả cũng được sử dụng để khai thác SFUPs thay thế cấu trúc UP-tree được sử dụng trong thuật toán SKYMINE. Các kết quả thực nghiệm cũng cho thấy thuật toán SkyFUP đề xuất hiệu quả hơn so với thuật toán SKYMINE được đề xuất trước đó.

### 3. Các định nghĩa và ký hiệu

Cho $I = \{x_1, x_2, \ldots, x_m\}$ là một tập hợp $m$ mặt hàng (item) khác nhau trong cơ sở dữ liệu giao dịch $D = \{T_1, T_2, \ldots, T_n\}$ có $n$ giao dịch (transaction), với $\forall\, T_j \in D$, $T_j = \{x_l | l = 1, 2, \ldots, N_j, x_l \in I\}$, trong đó $N_j$ là số lượng các mặt hàng trong giao dịch $T_j$. Mỗi mặt hàng $x_i$ trong $I$ có một giá trị lợi nhuận (profit value) là $Prf(x_i)$. Mỗi mặt hàng $x_i$ trong giao dịch $T_j$ có số lượng mua là $Pq(x_i, T_j)$. Một ví dụ về cơ sở dữ liệu giao dịch $D$ được cho trong Bảng 1 và lợi nhuận của các mặt hàng được cho trong Bảng 2.

**Bảng 1:** *Cơ sở dữ liệu giao dịch D*

| TID | Giao dịch ($T$) | Số lượng mua ($Pq$) | Độ hữu ích ($U$) | Độ hữu ích giao dịch ($TU$) |
|---|---|---|---|---|
| $T_1$ | $a, b, c, d$ | 2, 2, 7, 10 | 8, 6, 7, 20 | 41 |
| $T_2$ | $a, b, e, f$ | 1, 1, 2, 2 | 4, 3, 6, 2 | 15 |
| $T_3$ | $a, e$ | 3, 6 | 12, 18 | 30 |
| $T_4$ | $b, c, d$ | 2, 5, 2 | 6, 5, 4 | 15 |
| $T_5$ | $a, b, e$ | 5, 3, 1 | 20, 9, 3 | 32 |
| $T_6$ | $a, b, c, d, e, f$ | 1, 1, 4, 2, 5, 2 | 4, 3, 4, 4, 15, 2 | 32 |
| $T_7$ | $c, d$ | 4, 5 | 4, 10 | 14 |
| $T_8$ | $b, c, d, e$ | 2, 1, 2, 3 | 6, 1, 4, 9 | 20 |

**Bảng 2:** *Lợi nhuận của các mặt hàng*

| Mặt hàng | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|
| Lợi nhuận | 4 | 3 | 1 | 2 | 3 | 1 |

Tần số xuất hiện của một tập mặt hàng $X$ trong $D$, ký hiệu: $SUP(X)$, là số lượng các giao dịch $T_j$ trong cơ sở dữ liệu $D$ có chứa $X$. Ví dụ: Trong Bảng 1, $SUP(b) = 6$ và $sup(b, c, d) = 4$.

Độ hữu ích của một mặt hàng $x_i$ trong một giao dịch $T_j$, ký hiệu: $U(x_i, T_j)$. Ví dụ: Độ hữu ích của mặt hàng $\{b\}$ trong giao dịch $T_1$ được tính: $u(b, T_1) = Prf(b) * Pq(b, T1) = 3 * 2 = 6$. Độ hữu ích của một tập mặt hàng $X = \{x_1, x_2, \ldots, x_k\}$ trong giao dịch $T_j$, ký hiệu: $U(X, T_j)$ và được định nghĩa: $U(X, T_j) = \sum_{x_i \in X} U(x_i, T_j)$. Ví dụ: $U(d, T_1) = Prf(d) * Pq(d, T_1) = 2 * 10 = 20$ và $U(bcd, T_1) = 33$. Độ hữu ích của một tập mặt hàng $X$ trong cơ sở dữ liệu giao dịch $D$, ký hiệu: $U(X)$ và được định nghĩa: $U(X) = \sum_{X \subseteq T_j \wedge T_j \in D} U(X, T_j)$. Ví dụ: $U(a) = U(a, T_1) + U(a, T_2) + U(a, T_3) + U(a, T_5) + U(a, T_6) = 48$ *và* $U(bcd) = 70$. Độ hữu ích của một giao dịch $T_j$ trong cơ sở dữ liệu $D$, ký hiệu: $TU(T_j)$ và được định nghĩa: $TU(T_j) = \sum_{x_i \in T_j} U(x_i, T_j)$. Ví dụ: $TU(T_1) = U(a, T_1) + U(b, T_1) + U(c, T_1) + U(d, T_1) = 41$ và $TU(T_2) = 15$.

Độ hữu ích trọng số giao dịch của một tập mặt hàng $X$ trong cơ sở dữ liệu $D$ được kí hiệu là $TWU(X)$ và được định nghĩa: $TWU(X) = \sum_{X \subseteq T_j \in D} TU(T_j)$. Ví dụ: $TWU(ac) = TU(T_1) + TU(T_6) = 73$ và $TWU(cd) = 108$.

Một thứ tự toàn phần $\prec$ được xây dựng dựa trên việc sắp xếp tăng dần theo $TWU$ của các mặt hàng trong cơ sở dữ liệu $D$. Trong cơ sở dữ liệu đã cho ở Bảng 1, thứ tự sắp toàn phần các mặt hàng *là:* $f \prec c \prec d \prec e \prec a \prec b$. Bảng 3 thể hiện $TWU$ của các mặt hàng sau khi sắp tăng dần và Bảng 4 thể hiện cơ sở dữ liệu $D$ sau khi sắp tăng dần theo $TWU$.

**Bảng 3:** *$TWU$ của các mặt hàng sau khi được sắp tăng dần*

| Items | $f$ | $c$ | $d$ | $e$ | $a$ | $b$ |
|---|---|---|---|---|---|---|
| twu | 47 | 122 | 122 | 129 | 150 | 155 |

**Bảng 4:** *Cơ sở dữ liệu sau khi sắp tăng dần theo TWU*

| **TID** | **Giao dịch ($T$)** | **Số lượng mua ($Pq$)** | **Độ hữu ích ($U$)** | **Độ hữu ích giao dịch ($TU$)** |
|---|---|---|---|---|
| $T_1$ | $c, d, a, b$ | 7, 10, 2, 2 | 7, 20, 8, 6 | 41 |
| $T_2$ | $f, e, a, b$ | 2, 2, 1, 1 | 2, 6, 4, 3 | 15 |
| $T_3$ | $e, a$ | 6, 3 | 18, 12 | 30 |
| $T_4$ | $c, d, b$ | 5, 2, 2 | 5, 4, 6 | 15 |
| $T_5$ | $e, a, b$ | 1, 5, 3 | 3, 20, 9 | 32 |
| $T_6$ | $f, c, d, e, a, b$ | 2, 4, 2, 5, 1, 1 | 2, 4, 4, 15, 4, 3 | 32 |
| $T_7$ | $c, d$ | 4, 5 | 4, 10 | 14 |
| $T_8$ | $c, d, e, b$ | 1, 2, 3, 2 | 1, 4, 9, 6 | 20 |

Tập tất cả các mặt hàng sau $X$ trong $T_j$ được ký hiệu là $T_j|X$. Ví dụ: Trong Bảng 4, $T_2|\{f, a\} = \{b\}$ và $T_1|\{d\} = \{a, b\}$. Độ hữu ích sau tập mặt hàng $X$ trong một giao dịch $T_j$, ký hiệu: $RU(X, T_j)$, là tổng độ hữu ích của tất cả các mặt hàng sau $X$ trong $T_j$, và được định nghĩa: $RU(X, T_j) = \sum_{x_i \in (T_j|X)} U(x_i, T_j)$. Ví dụ: $RU(d, T_1) = U(a, T_1) + U(b, T_1) = 14$.

Để xem xét cùng lúc cả hai yếu tố tần suất và độ hữu ích, các định nghĩa về khai thác tập phổ biến hữu ích cao được trình bày bên dưới.

**Định nghĩa 1.** Một tập mặt hàng $X$ thống trị/vượt trội một tập mặt hàng $Y$ trong $D$ khi và chỉ khi $SUP(X) \geq SUP(Y)$ và $U(X) > U(Y)$ hoặc $SUP(X) > SUP(Y)$ và $U(X) \geq U(Y)$, ký hiệu: $X \rhd Y$.

Ví dụ: Trong Bảng 1, xét tập mặt hàng $\{b\}$ và $\{a, c\}$. Ta có $SUP(b) = 6$, $SUP(a, c) = 2$, $U(b) = 33$, $U(a, c) = 23$. Do đó, $\{b\} \rhd \{a, c\}$ vì $SUP(b) > SUP(ac)$ và $U(b) > U(ac)$. Tương tự, tập mặt hàng $\{b, c\} \rhd \{f, e, b\}$ vì $SUP(bc) > SUP(feb)$ và $U(bc) > U(feb)$.

**Định nghĩa 2.** Một tập mặt hàng $X$ trong một cơ sở dữ liệu $D$ là một tập hữu ích phổ biến thuộc đường chân trời (Skyline Frequent-Utility Pattern - SFUP) khi và chỉ khi $X$ không bị thống trị bởi bất kỳ một tập mặt hàng nào khác trong cơ sở dữ liệu về cả tần suất và độ hữu ích (nghĩa là không tồn tại bất kỳ một tập mặt hàng $Y \neq X$ nào thỏa điều kiện: $SUP(Y) > SUP(X)$ và $U(Y) > U(X)$.

Ví dụ: Trong Bảng 1, tần số và độ hữu ích của $\{b\}$ được tính lần lượt là 6 và 33; tần số và độ hữu ích của $\{c,d\}$ được tính lần lượt là 5 và 63; tần số và độ hữu ích của $\{e,a\}$ được tính lần lượt là 4 và 82. Các tập tập mặt hàng $\{b\}$, $\{c,d\}$ và $\{e,a\}$ được xem là các $SFUP$ vì không tồn tại bất kỳ tập mặt hàng nào khác thống trị chúng về cả tần số và độ hữu ích (nghĩa là không có tập mặt hàng nào có cả tần số và độ hữu ích lớn hơn những tập mặt hàng này).

### 4. Thuật toán SkyMiner

Trong phần này, chúng tôi đề xuất một thuật toán khai thác tập hữu ích cao phổ biến thuộc đường chân trời. Thuật toán 1 (SkyMiner) là thuật toán chính, có dữ liệu đầu vào là cơ sở dữ liệu giao dịch $D$, dữ liệu ra là tập Skyline Frequent-Utility Patterns ($SFUPs$). Khởi đầu quét cơ sở dữ liệu $D$ để tính $TWU(i)$ cho mỗi mục $i$ có trong tập mục $I$ trình bày ở dòng 1. Dòng 2 sắp xếp các mặt hàng trong tập $I$ tăng dần theo giá trị của $TWU$, đồng thời sắp xếp các mục của tất cả các giao dịch trong $D$ theo thứ tự của tập $I$. Từ dòng 3 đến dòng 8 khởi tạo danh sách *utility-list* [24] một phần tử, khởi tạo cấu trúc $EUCS$ [12], khởi tạo cấu trúc $Umax$ [21] và danh sách kết quả $SFUPs$ để chứa các phần tử là SFUP. Dòng 9 gọi thực hiện Thuật toán 2 (SearchSFUP). Dòng 10 trả về kết quả là tập $SFUPs$ và kết thúc thuật toán.

**Thuật toán 1: SkyMiner**

**Vào**: Cơ sở dữ liệu giao dịch $D$.
**Ra**: Tập tất cả các mục Skyline Frequent-Utility Parttens (SFUPs).
1. Quét cơ sở dữ liệu $D$ để tính $TWU(i)$ cho mỗi mục $i$ có trong $I$.
2. Sắp xếp tập $I$ tăng theo $TWU$, sắp xếp các mục của tất cả các giao dịch trong $D$ theo thứ tự của tập $I$.
3. Khởi tạo danh sách *utility-list* một phần tử là $ULs$
4. Khởi tạo cấu trúc $EUCS$
5. **for** each $k$ ($1 \leq k \leq |D|$) **do**
6. $Umax(k) = 0$;
7. **end for**
8. SFUPs $\leftarrow \emptyset$;
9. *SearchSFUP*$(null, ULs, Umax, SFUPs)$
10. return $SFUPs$

Thuật toán 2 (SearchSFUP) có dữ liệu đầu vào gồm có $P$: *utility-list* với vai trò là tiền tố; $ULs$: Danh sách các *utility-list* có tiền tố là $P$; $Umax$: Danh sách các độ hữu ích lớn nhất theo từng độ hỗ trợ tại thời điểm thủ tục này được gọi; $SFUPs$: các tập mặt hàng là $SFUP$ dự kiến tại thời điểm đang xét. Dữ liệu ra là tập $SFUPs$ được cập nhật. Dòng 1 duyệt qua từng *utility-list* có trong danh sách *utility-list* một phần tử $ULs$. Dòng 2 xác định giá trị $Umax(k)$ với $k = SUP(X)$ và đặt giá trị này là $maxUtil$. Dòng 3, 4, 5 kiểm tra điều kiện nếu $U(X) \geq maxUtil$ thì gọi thủ tục $UpdateSFUP$ để cập nhật tập $SFUPs$. Dòng 6 áp dụng chiến lược tỉa *U-Prune* [24] bằng cách kiểm tra điều kiện, nếu $U(X) + RU(X) \geq maxUtil$ thì thực hiện các bước tiếp theo để tạo danh sách *utility-list* mở rộng từ *utility-list* $X$, ngược lại ngừng mở rộng với $X$. Dòng 9 áp dụng chiến lược tỉa *EUCS-Prune*, nếu $EUCS(X,Y) \geq maxUtil$ thì thực hiện thủ tục $\mathrm{Construct}(P,X,Y)$ để tạo *uility-list* $XY$ từ 2 *utility-list* $X$ và $Y$ và thêm vào danh sách $exULs$, ngược lại không tạo *utility-list* $XY$. Dòng 13 gọi đệ quy thủ tục SearchSFUP để tiếp tục mở rộng tập SFUPs.

**Thuật toán 2: SearchSFUP**

**Vào**: $P$: *utility-list* với vai trò là tiền tố;
$ULs$: Danh sách các *utility-list* có tiền tố là *utility-list P*
$Umax$: Danh sách các độ hữu ích lớn nhất theo từng độ hỗ trợ tại thời điểm khai thác.
$SFUPs$: Tập các mục là SFUP dự kiến tại thời điểm đang xét.
**Ra**: Tập $SFUPs$ được cập nhật.
1. **for** each $X \in ULs$ **do**
2. $maxUtil = Umax(SUP(X))$;
3. **if** $U(X) \geq maxUtil$ **then**
4. $UpdateSFUP(X, SFUPs, Umax)$;
5. **end if**
6. **if** $U(X) + RU(X) \geq maxUtil$ //áp dụng chiến lược tỉa U-Prune
7. $exULs = \emptyset$ //Khởi tạo danh sách utility-list mở rộng từ X
8. **for** each $Y$ after $X$ in $ULs$ **do**
9. **if** $EUCS(X, Y) \geq maxUtil$ **then** //Áp dụng chiến lược tỉa EUCP
10. $exULs \leftarrow \text{Construct}(P, X, Y)$;
11. **end if**
12. **end for**
13. $SearchSFUP(X,\ exULs,\ Umax,\ SFUPs)$; //gọi đệ quy thuật toán.
14. **end if**
15. **end for**

Thuật toán 3 (UpdateSFUP) với dữ liệu vào gồm $X$: *utility-list* đang xem xét có khả năng là một $SFUP$; $SFUPs$: Tập các mục skyline frequent utility parttens dự kiến tại thời điểm đang xét; $Umax$: Danh sách các tiện ích lớn nhất theo từng độ hỗ trợ. Dữ liệu ra là tập $SFUPs$ và danh sách $Umax$ được cập nhật. Dòng 1, 2 tìm tập $Y \in SFUPs$ sao cho $SUP(Y) \geq SUP(X)$, nếu không tồn tại tập $Y$ nào ($Y = null$) hoặc có tập $Y$ và $U(X) > U(Y)$ thì thực hiện cập nhật $SFUPs$ ở các bước tiếp theo. Từ dòng 3 đến dòng 7, tìm các phần tử $Z \in SFUPs$ sao cho $SUP(X) \geq SUP(Z)$ và $U(X) \geq U(Z)$ thì xóa Z khỏi $SFUPs$. Dòng 8 đến dòng 12 cập nhật danh sách $Umax$ từ vị trí 1 đến vị trí $k = SUP(X)$, nếu $Umax(k) < U(X)$ thì cập nhật $Umax(k) = U(X)$. Dòng 13 thêm $X$ vào danh sách $SFUPs$.

**Thuật toán 3: UpdateSFUP**

**Vào**: $X$: *utility-list*;
$SFUPs$: Tập các mục skyline frequent utility parttens cần cập nhật.
$Umax$: Danh sách các độ hữu ích lớn nhất theo từng độ hỗ trợ.
**Ra**: Tập $SFUPs$ được cập nhật.
Danh sách $Umax$ được cập nhật.
1. Tìm tập $Y \in SFUPs$ sao cho $SUP(Y) \geq SUP(X)$
2. **if** $Y = null\ or\ U(X) > U(Y)$ **then**
3. **for** each $Z \in SFUPs$ **do**
4. **if** $SUP(X) \geq SUP(Z)\ and\ U(X) \geq U(Z)$ **then**
5. Xóa Z khỏi $SFUPs$

6. **end if**
7. **end for**
8. **for** $k = SUP(X)\ downto\ 1$ **do**
9. **if** $Umax(k) < U(X)$ **then**
10. $Umax(k) = U(X)$;
11. **end if**
12. **end for**
13. $SFUPs \leftarrow X$; // $X$ được chuyển thành $SFUP$, sau đó đưa X vào tập SFUPs
14. **end if**

Thuật toán 4 (Construct) thực hiện kết hợp 2 *utility-list* $Px$ và $Py$ thành *utility-list* $Pxy$. Dòng 1 khởi tạo giá trị ban đầu cho $ULA$ là $U(P) + RU(P)$. Dòng 2, 3 duyệt qua từng phần tử $ex \in Px$ và tìm phần tử $ey \in Py$ sao cho $ex.tid = ey.tid$. Nếu tìm thấy thì tạo phần tử $exy$ kết hợp từ $ex$ và $ey$ trong các trường hợp xem xét tại dòng 4. Nếu *utility-list* tiền tố $P \neq \emptyset$ (trường hợp 1), nghĩa là *utility-list* $Pxy$ đang tạo tương ứng với tập mặt hàng có từ 3 mặt hàng trở lên, ngược lại (trường hợp 2) thì $Pxy$ là *utility-list* tương ứng với tập mặt hàng có 2 mặt hàng. Dòng 6 tạo phần tử $exy$ ứng với trường hợp 1, dòng 8 tạo phần tử $exy$ ứng với trường hợp 2, dòng 10 thêm $exy$ vào *utility-list* $Pxy$. Dòng 12 xét điều kiện nếu không tồn tại $ey \in Py$ mà $ex.tid = ey.tid$ thì áp dụng chiến lược tỉa LA-Prune [29] từ dòng 13 đến 17. Dòng 20 trả về kết quả là *utility-list* $Pxy$.

**Thuật toán 4: Construct**

**Vào**: $P$: *utility-list* với vai trò là tiền tố.
$Px,\ Py$: Hai *utility-list* cần kết hợp.
$maxUtil$: Tiện ích lớn nhất tại độ hỗ trợ tương ứng với $Px$.
**Ra**: $Pxy$: *utility-list* sau khi kết hợp $Px$ và $Py$.
1. $Set\ ULA = U(P) + RU(P)$;
2. **for** each element $ex \in Px$ **then**
3. **if** $\exists\ ey \in Py \wedge ex.tid = ey.tid$ **then**
4. **if** $P \neq \emptyset$ **then**
5. Tìm $e \in P$ sao cho $e.tid = ex.tid$;
6. $exy =< ex.tid,\ ex.U + ey.U - e.U,\ ey.RU >$;
7. **else**
8. $exy =< ex.tid,\ ex.U + ey.U,\ ey.RU >$;
9. **end if**
10. $Pxy \leftarrow exy$;
12. **else**
13. $ULA = ULA - ex.U - ex.RU$;
14. **if** $ULA < maxUtil$ **then** // áp dụng chiến lược tỉa LA-Prune
15. return null;
16. **end if**
17. Continue;
18. **end if**
19. **end for**
20. return $Pxy$;

## 5. Thực nghiệm

Thuật toán *SkyMiner* được cài đặt bằng ngôn ngữ lập trình Java, trên máy tính Dell Precision Tower 3620, Intel Core i7-7800X CPU @3.5GHz, bộ nhớ RAM 32GB trên hệ điều hành Windows 10. Các cơ sở dữ liệu thử nghiệm gồm Chess, Mushroom, Accident, Foodmart, Retail, Chainstore được tải từ thư viện SPMF [30] và cơ sở dữ liệu tổng hợp có tên là T10I4N4KD500K [31]. Chi tiết các cơ sở dữ liệu được trình bày trong Bảng 5. Thực nghiệm của thuật toán *SkyMiner* được so sánh với thuật toán mới nhất cùng khai thác tập SFUPs là *SkyFUP* [21]. Kết quả thực nghiệm được đánh giá dựa trên thời gian thực thi, dung lượng bộ nhớ sử dụng và số lượng ứng viên tạo ra trong quá trình thực thi của 2 thuật toán.

**Bảng 5:** *Đặc điểm các cơ sở dữ liệu thực nghiệm*

| Cơ sở dữ liệu | Số lượng giao dịch | Số lượng mặt hàng (I) | Độ dài trung bình (A) | Độ dày (A/I) % |
|---|---|---|---|---|
| Chess | 3,196 | 75 | 37 | 49.3333 |
| Mushroom | 8,124 | 119 | 23 | 19.3277 |
| Accident | 340,183 | 468 | 33.8 | 7.2222 |
| Foodmart | 4141 | 1559 | 4.4 | 0.2822 |
| Retail | 88,162 | 16,470 | 10.3 | 0.0625 |
| Chainstore | 1,112,949 | 46,086 | 7.3 | 0.0158 |
| T10I4N4KD500K | 500,000 | 3547 | 10 | 0.0028 |

### *5.1. So sánh thời gian thực thi*

Thời gian thực thi của 2 thuật toán trên các cơ sở dữ liệu thử nghiệm được trình bày trong Bảng 6. Kết quả cho thấy thời gian thực thi của thuật toán *SkyMiner* nhanh hơn thuật toán *SkyFUP* ở tất cả các bộ dữ liệu thử nghiệm. Với các cơ sở dữ liệu dày như *Chess*, *Mushroom* và *Accident* thì thời gian thực thi của thuật toán *SkyMiner* nhanh hơn thuật toán *SkyFUP* nhưng không đáng kể do 2 chiến lược tỉa *EUCS-Prune* và *LA-Prune* không loại bỏ được nhiều ứng viên. Tuy nhiên, đối với các cơ sở dữ liệu thưa như *Foodmart*, *Retail*, *Chainstore* và *T10I4N4KD500K* thì thời gian thực thi của thuật toán *SkyMiner* nhanh hơn nhiều so với thuật toán *SkyFUP*. Cụ thể, với cơ sở dữ liệu *Foodmart*, thuật toán *SkyMiner* có thời gian thực thi là 0.03 giây, nhanh hơn 10 lần so với thuật toán *SkyMiner* là 0.31 giây. Đặc biệt, với cơ sở dữ liệu *Chainstore*, thời gian thực thi của thuật toán *SkyMiner* là 24.01 giây, nhanh hơn 88 lần so với thuật toán *SkyFUP*.

Để so sánh thời gian thực hiện theo độ lớn của cùng một cơ sở dữ liệu, chúng tôi thực nghiệm trên cơ sở dữ liệu T10I4N4KD|X|K có độ lớn |X| tăng trưởng từ 100,000 giao dịch đến 500,000 giao dịch. Kết quả trình bày ở Hình 3a cho thấy thời gian thực hiện của thuật toán *SkyMiner* chỉ tăng nhẹ ở mức thấp từ 1.37 giây (|X| = 100,000) đến 4.72 giây (|X| = 500,000). Trong khi đó, thuật toán *SkyFUP* có thời gian thực hiện tăng tuyến tính mạnh theo độ lớn từ 49.6 giây (|X| = 100,000) đến 335.7 giây (|X| = 500,000). Kết quả này cho thấy với cùng một cơ sở dữ liệu thì thuật toán SkyFUP kém hiệu quả khi số lượng giao dịch tăng cao, trong khi hiệu suất thực thi của thuật toán SkyMiner vẫn ổn định.

**Bảng 6:** *So sánh thời gian thực thi của hai thuật toán SkyFUP và SkyMiner*

| Đơn vị (giây) | SkyFUP | SkyMiner |
|---|---|---|
| Chess | 33.37 | 31.82 |
| Mushroom | 1.58 | 1.41 |
| Accident | 985.61 | 969.98 |
| Foodmart | 0.31 | 0.03 |
| Retail | 78.47 | 2.69 |
| Chainstore | 2,128.55 | 24.01 |
| T10I4N4KD500K | 334.71 | 4.72 |

### *5.2. So sánh bộ nhớ sử dụng*

Bảng 7 trình bày bộ nhớ sử dụng của 2 thuật toán. Với các cơ sở dữ liệu dày như *Chess*, *Mushroom* và *Accident* có số lượng mặt hàng tương đối thấp (Bảng 8) nên việc khởi tạo cấu trúc EUCS (chiến lược 3) không ảnh hưởng nhiều đến bộ nhớ sử dụng của thuật toán *SkyMiner*. Do đó, dung lượng bộ nhớ sử dụng của thuật toán *SkyMiner* thấp hơn thuật thuật toán *SkyFUP*. Đặt biệt hơn, các cơ sở dữ liệu có độ dày trung bình như *Mushroom* và *Accident* thì dung lượng bộ nhớ sử dụng của thuật toán *SkyMiner* thấp hơn khoảng 2 lần so với thuật toán *SkyFUP*. Các cơ sở dữ liệu thưa như *Foodmart*, *Retail*, *Chainstore* và *T10I4N4KD500K* thì bộ nhớ sử dụng của *SkyMiner* hiệu quả hơn thuật toán *SkyFUP* ở các cơ sở dữ liệu có số lượng mặt hàng thấp như *Foodmart* (1559 mặt hàng) và *T10I4N4KD500K* (3547 mặt hàng), trong khi đó, thuật toán SkyMiner kém hiệu quả hơn thuật toán *SkyFUP* trên các cơ sở dữ liệu có số lượng mặt hàng rất lớn như Retail (16,470 mặt hàng) và Chainstore (46,086 mặt hàng), kết quả này cho thấy việc áp dụng chiến lược tỉa *EUCS-Prune* sẽ sử dụng nhiều bộ nhớ hơn đối với các cơ sở dữ liệu có số lượng mặt hàng lớn. Với cơ sở dữ liệu T10I4N4KD|X|K, thuật toán *SkyMiner* sử dụng bộ nhớ thấp hơn thuật toán *SkyFUP* ở tất cả các ngưỡng độ lớn |X| từ 100,000 đến 500,000 (Hình 3b). Kết quả này cho thấy với cơ sở dữ liệu thưa có số lượng mặt hàng trung bình thì thuật toán SkyMiner luôn sử dụng bộ nhớ ít hơn thuật toán SkyFUP, không phụ thuộc vào độ lớn cơ sở dữ liệu.

**Bảng 7:** *So sánh bộ nhớ sử dụng của hai thuật toán*

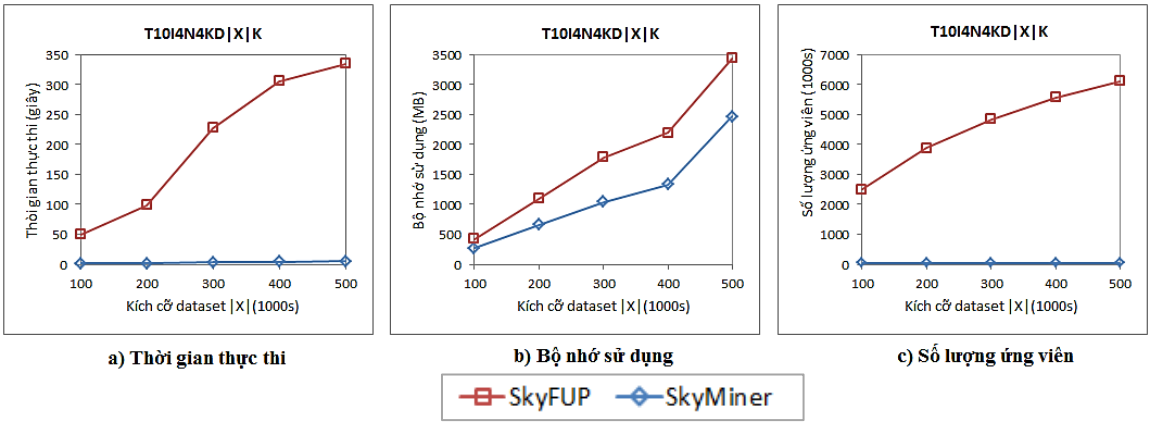
| Đơn vị (MB) | SkyFUP | SkyMiner |
|---|---|---|
| Chess | 2,043 | 2,042 |
| Mushroom | 1,366 | 725 |
| Accident | 5,624 | 2,796 |
| Foodmart | 1,488 | 756 |
| Retail | 1,579 | 2,114 |
| Chainstore | 2,766 | 10,294 |
| T10I4N4KD500K | 3,440 | 2,462 |

### *5.3. So sánh số lượng ứng viên*

**Bảng 8:** *So sánh số lượng ứng viên phát sinh của hai thuật toán*

| Đơn vị (ứng viên) | SkyFUP | SkyMiner |
|---|---|---|
| Chess | 3,397,909 | 2,552,448 |
| Mushroom | 40,949 | 32,565 |
| Accident | 2,555,599 | 1,889,989 |
| Foodmart | 39,721 | 2,190 |
| Retail | 5,158,951 | 857,461 |
| Chainstore | 30,426,801 | 1,072,259 |
| T10I4N4KD500K | 6,108,502 | 50,534 |

Bảng 8 trình bày số lượng ứng viên tạo ra trong quá trình khai thác *SFUP* của hai thuật toán trên tập các cơ sở dữ liệu thực nghiệm. Kết quả cho thấy với các cơ sở dữ liệu dày như *Chess*, *Mushroom* và *Accident* thì thuật toán *SkyMiner* tỉa ứng viên tốt hơn thuật toán *SkyFUP*, tuy nhiên, sự chênh lệch là không đáng kể. Với các cơ sở dữ liệu thưa như *Foodmart*, *Retail*, *Chainstore* và *T10I4N4KD500K*, thuật toán *SkyMiner* cho thấy hiệu quả cắt tỉa ứng viên vượt trội so với thuật toán *SkyFUP*. Số lượng ứng viên do thuật toán *SkyMiner* sinh ra ít hơn từ 6 đến 120 lần so với thuật toán *SkyFUP*. Với cơ sở dữ liệu T10I4N4KD|X|K, kích thước cơ sở dữ liệu tăng từ 100,000 đến 500,000 thì số lượng ứng viên sinh ra của thuật toán *SkyMiner* ít hơn nhiều so với thuật toán *SkyFUP* (Hình 3c). Cụ thể, thuật toán *SkyMiner* sinh ra từ 45,200 đến 50,534 ứng viên, trong khi số lượng ứng viên sinh ra của thuật toán *SkyFUP* từ 2,497,428 đến 6,108,502. Kết quả này chứng tỏ rằng các chiến lược tỉa được áp dụng trong thuật toán *SkyMiner* đã làm giảm số lượng ứng viên một cách đáng kể, từ đó tăng hiệu suất thực thi của thuật toán, đặt biệt với hai chiến lược tỉa *LA-Prune* và *EUCS-Prune*.




**a) Thời gian thực thi** **b) Bộ nhớ sử dụng** **c) Số lượng ứng viên**

**Hình 3:** *So sánh hiệu suất thực thi trên cơ sở dữ liệu T10I4N4KD|X|K*

## 6. Kết luận

Bài báo này đề xuất thuật toán *SkyMiner* để khai thác SFUPs trên cơ sở dữ liệu giao dịch dựa trên độ hữu ích và tần suất xuất hiện của tập mặt hàng. Cấu trúc *utility-list*

được sử dụng để tổ chức, lưu trữ dữ liệu trong quá trình khai thác. Ngoài chiến lược tỉa *U-Prune* đã được áp dụng trong các thuật toán khai thác tập SFUPs trước đây, chúng tôi áp dụng thêm hai chiến lược tỉa là *LA-Prune* và *EUCS-Prune* để tăng hiệu suất thực thi của thuật toán. Kết quả thực nghiệm cho thấy thuật toán *SkyMiner* cho kết quả tốt hơn thuật toán mới nhất là *SkyFUP* về thời gian thực thi, bộ nhớ sử dụng và số lượng ứng viên sinh ra trong quá trình khai thác.

Hướng phát triển tiếp theo là cải tiến cấu trúc dữ liệu lưu trữ để tăng hiệu suất thực thi của thuật toán trên các cơ sở dữ liệu dày, khai thác SFUPs trên các dạng cơ sở dữ liệu khác như cơ sở dữ liệu tăng trưởng (increamental database), cơ sở dữ liệu động (dynamic database).

## TÀI LIỆU THAM KHẢO

[1] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM Sigmod Record,* Vol. 29, No. 2, pp. 1-12, 2000.

[2] G. Grahne and J. Zhu, "Fast algorithms for frequent itemset mining using FP-Trees," *IEEE Transactions on Knowledge and Data Engineering,* Vol. 17, No. 10, pp. 1347-1362, 2005.

[3] B. Vo, T. Le, T. P. Hong, and B. Le, "Fast updated frequent-itemset lattice for transaction deletion," *Data & Knowledge Engineering*, Vol. 96, pp. 78-89, 2015.

[4] Z. H. Deng and S. L. Lv, "Fast mining frequent itemsets using Nodesets," *Expert Systems with Applications*, Vol. 41, No. 10, pp. 4505-4512, 2014.

[5] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," In *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, pp. 487-499, 1994.

[6] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules," *Knowledge Discovery and Data Mining*, pp. 283-286, 1997.

[7] B. Vo, T. P. Hong, and B. Le, "A lattice-based approach for mining most generalization association rules," *Knowledge-Based Systems*, Vol. 45, pp. 20-30, 2013.

[8] Y. Liu, W. K. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," *In Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 689-695, 2005.

[9] H. Yao and H. J. Hamilton, "Mining itemsets utilities from transaction databases," *Data and Knowledge Engeneering,* Vol. 59, No. 3, pp. 603-626, 2006.

[10] V. S. Tseng, C. W. Wu, B. E. Shie, and P. S. Yu, "UP-Growth: an efficient algorithm for high utility itemset mining," *In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 253-262, 2010.

[11] J. Liu, K. Wang, and B. C. Fung, "Direct discovery of high utility itemsets without candidate generation," *IEEE 12th International Conference on Data Mining*, pp. 984-989, 2012.

[12] P. Fournier-Viger, C. W. Wu, S. Zida, and V. S. Tseng, "FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning," *International Symposium on Methodologies for Intelligent Systems*, Vol. 8502, pp. 83-92, 2014.

[13] G. I. Webb, "Filtered-top-k association discovery," *Data Mining and Knowledge Discovery*, Vol. 1, No. 3, pp. 183-192, 2011.

[14] P. Fournier-Viger, C. W. Wu, and V. S. Tseng, "Mining top-k association rules," *Canadian Conference on Artificial Intelligence*, pp. 61-73, 2012.

[15] L. T. Nguyen, B. Vo, L. T. Nguyen, P. Fournier-Viger, and A. Selamat, "ETARM: an efficient top-k association rule mining algorithm," *Applied Intelligence,* Vol. 48, No. 5, pp. 1148-1160, 2018.

[16] V. S. Tseng, C. W. Wu, P. Fournier-Viger, and S. Y. Philip, "Efficient algorithms for mining top-k high utility itemsets," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, No. 1, pp. 54-67, 2015.

[17] K. Singh, S. S. Singh, A. Kumar, and B. Biswas, "TKEH: an efficient algorithm for mining top-k high utility itemsets," *Applied Intelligence*, Vol. 49, No. 3, pp. 1078-1097, 2019.

[18] S. Krishnamoorthy, "Mining top-k high utility itemsets with effective threshold raising strategies," *Expert Systems with Applications*, Vol. 117, pp. 148-165, 2019.

[19] V. Goyal, A. Sureka, and D. Patel, "Efficient skyline itemsets mining," *The International C* Conference on Computer Science & Software Engineering*, pp. 119-124, 2015.

[20] J. S. Pan, J. C. W. Lin, L. Yang, P. Fournier-Viger, and T. P. Hong, "Efficiently mining of skyline frequent-utility patterns," *Intelligent Data Analysis*, Vol. 21, No. 6, pp. 1407-1423, 2017.

[21] J.C.W. Lin, L. Yang, P. Fournier-Viger, and T.P. Hong, "Mining of skyline patterns by considering both frequent and utility constraints," *Engineering Applications of Artificial Intelligence*, Vol. 77, pp. 229-238, 2019.

[22] B. Le, H. Nguyen, and B. Vo, "An efficient strategy for mining high utility itemsets," *International Journal of Intelligent Information and Database Systems*, Vol.5, No. 2, pp. 164-176, 2011.

[23] V. S. Tseng, B. E. Shie, C. W. Wu, and S. Y. Philip, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE transactions on knowledge and data engineering,* Vol.25, pp. 1772-1786, 2012.

[24] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management,* pp. 55-64, 2012.

[25] S. Zida, P. Fournier-Viger, J. C. W. Lin, C. W. Wu, and V. S. Tseng, "EFIM: A fast and memory efficient algorithm for high-utility itemset mining," *Knowledge and Information Systems*, Vol. 51, No. 2, pp. 595-625, 2017.

[26] S. Krishnamoorthy, "HMiner: Efficiently mining high utility itemsets," *Expert Systems with Applications*, Vol. 90, pp. 168-183, 2017.

[27] J.S. Yeh, Y. C. Li, and C. C. Chang, "Two-phase algorithms for a novel utility-frequent mining model," *International Conference on Emerging Technologies in Knowledge Discovery and Data Mining*, pp. 433-444, 2007.

[28] V. Podpecan, N. Lavrac, and I. Kononenko, "A fast algorithm for mining utility-frequent itemsets," *International Workshop on Constraint-based Mining and Learning*, pp. 9-20, 2007.

[29] S. Krishnamoorthy, "Pruning strategies for mining high utility itemsets," *Expert Systems with Applications*, Vol. 42, No. 5, pp. 2371-2381, 2015.

[30] P. Fournier-Viger, A. Gomariz, A. Soltani, and H. Lam, "An Open-Source Data Mining Library," 2014. [Online]. http://www.philippe-fournier-viger.com

[31] R. Agrawal and R. Srikant, "Quest synthetic data generator - IBM Almaden Research Center," 1994. [Online]. http://www.Almaden.ibm.com/cs/quest/syndata.html

## SUMMARY

## AN EFFICIENT ALGORITHM TO EXTRACT SKYLINE ITEMSETS

**Nguyen Thi Thanh Thuy, Manh Thien Ly,**
**Nguyen Van Le, Vu Van Vinh**
*Ho Chi Minh City University of Food Industry*
Received on 06/10/2020, accepted for publication on 23/12/2020

Mining skyline frequent-utility patterns (SFUPs) is the discovery of itemsets that surpasses all other itemsets in both frequency and utility in transactional database. The discovery of these itemsets is important for managers in finding items that customers buy many times and bring high profits for businesses. In recent years, there have been many algorithms proposed to exploit skyline frequent-utility patterns, of which SKYFUP-D is the most efficient algorithm. However, this algorithm still has limitations in both execution time and storage space. In this paper, we propose an effective method to exploit SFUPs faster by applying pruning strategies to reduce the number of candidates. Experimental results show that the execution time and storage space are significantly improved.

**Key words:** SFUPs; skyline frequent-utility itemset; EUCS; LA.