TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

HUỲNH PHÚC NGUYÊN – 52000091
MA NHẬT BIỂN – 52000015

NGHIÊN CỨU KHAI PHÁ K TẬP PHỔ BIẾN TIỆN ÍCH CAO ĐẦU TIÊN TỪ CƠ SỞ DỮ LIỆU KHÔNG CHẮC CHẮN

KHÓA LUẬN TỐT NGHIỆP

CHUYÊN NGÀNH KỸ THUẬT PHẦN MỀM

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

HUỲNH PHÚC NGUYÊN – 52000091
MA NHẬT BIỂN – 52000015

NGHIÊN CỨU KHAI PHÁ K TẬP PHỔ BIẾN TIỆN ÍCH CAO ĐẦU TIÊN TỪ CƠ SỞ DỮ LIỆU KHÔNG CHẮC CHẮN

KHÓA LUẬN TỐT NGHIỆP

CHUYÊN NGÀNH KỸ THUẬT PHẦN MỀM

Người hướng dẫn
Nguyễn Chí Thiện

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn Khoa Công nghệ thông tin đã tạo điều kiện cho chúng em được tiếp cận và hoàn thành bài báo cáo. Chúng em xin chân thành cảm ơn thầy Nguyễn Chí Thiện đã hướng dẫn hoàn thành bài báo cáo. Những sự hướng dẫn, quan tâm tận tình trong quá trình thực hiện báo cáo của thầy chính là động lực và là tiền đề để chúng em có được thành quả ngày hôm nay.

Trong quá trình làm bài báo cáo, do kiến thức cũng như kinh nghiệm còn nhiều hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, chúng em rất mong nhận được ý kiến đóng góp của thầy để chúng em có thể học hỏi được nhiều kĩ năng và kinh nghiệm, từ đó có được nền tảng để phát triển hơn cho các dự án sau.

Chúng em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 24 tháng 8 năm 2024

Tác giả

Huỳnh Phúc Nguyên

Ma Nhật Biển

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của thầy Nguyễn Chí Thiện. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 24 tháng 8 năm 2024

Tác giả

Huỳnh Phúc Nguyên

Ma Nhật Biển

NGHIÊN CỨU KHAI PHÁ K TẬP PHỔ BIẾN TIỆN ÍCH CAO ĐẦU TIÊN TỪ CƠ SỞ DỮ LIỆU KHÔNG CHẮC CHẮN

TÓM TẮT

Khai thác k tập phổ biến tiện ích cao đầu tiên từ bộ dữ liệu không chắc chắn là đề tài được đưa ra thảo luận trong thời gian gần đây và đã có nhiều cách tiếp cận được đưa ra để giải quyết bài toán. Tuy nhiên, những cách giải quyết thông thường đều gặp các vấn đề khi tìm kiếm quá nhiều mẫu tiện ích cao không chắc chắn do áp dụng các cách tiếp cận truyền thống.

Bài báo cáo này sẽ nghiên cứu về khai thác top-k tập mẫu phổ biến tiện ích cao từ bộ dữ liệu không chắc chắn. Ngoài việc khai thác các mẫu có tiện ích cao thì sẽ thêm tính phổ biến của mẫu dựa trên ngưỡng phổ biến. Trong nghiên cứu này, chúng tôi sẽ áp dụng hai giải pháp khai thác hiệu quả có tên TUHUFP và TUHUFP-Growth đi cùng với các chiến lược nhằm nâng cao hiệu suất thời gian và giảm thiểu không gian lưu trữ. Các thực nghiệm sẽ được tiến hành nhằm kiểm tra mức độ hiệu quả của giải pháp, gồm số lượng mẫu tham gia, thời gian khai thác và bộ nhớ được sử dụng.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC CHỮ VIẾT TẮT

expSup	Expected support
HUP	High utility pattern
TUHUFP	Top-k uncertain high utility frequent pattern
TID	Transaction identity
CUP	Candidate uncertain pattern
Prob	Probability
Uti	Utility
TWU	Transaction Weight Utilization
CPB	Conditional Pattern Base
UHUFP	Uncertain High Utility Frequent Pattern

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

Trong thời đại số hóa ngày nay, dữ liệu đã trở thành một nguồn tài nguyên vô cùng quý báu, đóng vai trò quan trọng trong mọi lĩnh vực của xã hội và kinh tế. Sự tích tụ lớn mạnh của dữ liệu từ các nguồn đa dạng như thiết bị di động, cảm biến, mạng xã hội và nhiều nguồn khác, tạo ra một biển số dữ liệu ngày càng phong phú và phức tạp. Khai thác dữ liệu (data mining) đã trở thành một công cụ quan trọng, giúp chúng ta phân tích, tìm kiếm mẫu, và trích xuất thông tin quan trọng từ lượng lớn dữ liệu này, từ đó doanh nghiệp và tổ chức có thể đưa ra tổng kết và dự đoán xu hướng, tìm ra các mối liên kết ẩn sau dữ liệu, và tạo ra giá trị thực sự từ thông tin

Một khía cạnh hấp dẫn của thách thức trong khai thác dữ liệu là việc trích xuất các mẫu thường xuyên hàng đầu (top-K frequent patterns) từ các cơ sở dữ liệu không chắc chắn, nơi tính không chắc chắn tự nhiên của dữ liệu tạo ra những phức tạp trong quá trình khai thác mẫu. Đồng thời, việc tìm kiếm thông tin giá trị mở rộng đến lĩnh vực các mẫu có tiện ích cao nhất hàng đầu (top-K high utility patterns), nhấn mạnh sự quan trọng của việc khám phá các mẫu đóng góp đáng kể vào tổng giá trị. Khác với các mẫu thường xuyên truyền thống, mẫu có tiện ích cao xem xét không chỉ tần suất xuất hiện mà còn giá trị liên quan đến mỗi mẫu. Giá trị này có thể bao gồm nhiều yếu tố, chẳng hạn như giá trị tiền tệ, điểm quan trọng hoặc các chỉ số quan trọng khác, tùy thuộc vào lĩnh vực ứng dụng.

Nghiên cứu này sẽ khám phá sâu sắc về những phức tạp khi khai thác các mẫu thường xuyên hàng đầu từ các cơ sở dữ liệu không chắc chắn và khám phá các mẫu có ích cao hàng đầu. Bằng cách phát triển các phương pháp tiên tiến được điều chỉnh để xử lý sự không chắc chắn và xem xét các yếu tố tiện ích, chúng tôi hướng đến việc khám phá các mẫu không chỉ thường xuyên mà còn có giá trị cao trong nhiều tình huống thực tế.

CHƯƠNG 2. CÔNG TRÌNH LIÊN QUAN

2.1 Khai thác tập mẫu phổ biến

Khai thác mẫu phổ biến là quá trình tìm kiếm và xác định các phần dữ liệu có mật độ xuất hiện cao và có liên hệ chặt chẽ với nhau. Các mẫu phổ biến này có thể xuất hiện trong nhiều lĩnh vực khác nhau, từ thị trường mua sắm đến nghiên cứu sinh học và phân tích dữ liệu mạng.

Đã có một số thuật toán nổi tiếng được thiết kế để khai thác mẫu phổ biến như Apriori [28], FP-growth [10], và Eclat [34]. Kết quả của khai thác mẫu phổ biến có thể được áp dụng rộng rãi, từ việc đề xuất sản phẩm cho đến dự đoán xu hướng thị trường, từ phân tích dữ liệu y tế đến phát hiện gian lận mạng.

2.2 Khai thác tập mẫu top-k và top-rank-k

Khai thác mẫu top-k và top-rank-k là quá trình tìm kiếm và xác định k mẫu quan trọng nhất từ một tập dữ liệu lớn. Trong trường hợp top-rank-k, mẫu được xác định dựa trên mức độ quan trọng của chúng như tần suất xuất hiện, giá trị thông tin, hoặc các yếu tố khác.

Đã có nhiều sáng kiến được đưa ra để giải quyết bài toán top-k và top-rank-k như sử dụng cấu trúc Node-list có tên NTK của Deng [6] hay các thuật toán iNTK algorithm [12] và BTK-algorithm [3]

2.3 Khai thác tập mẫu tiện ích cao

Khai thác tập mẫu có tiện ích cao là quá trình xác định và trích ra những mẫu dữ liệu đặc biệt có giá trị lớn hoặc quan trọng. Trong môi trường dữ liệu không chắc chắn, khi mỗi mẫu dữ liệu đi kèm với một độ không chắc chắn về giá trị, khai thác mẫu có giá trị cao trở nên cực kỳ hữu ích.

Đã có nhiều nghiên cứu được thực hiện để tìm các tập hợp HUIs. Điển hình như nghiên cứu của Liu và cộng sự, đã đưa ra thuật toán Two Phase [24] và mô hình TWU (Transaction Weight Utility) là phương pháp được đề xuất. Hay thuật toán HUP-Growth [22] sử dụng cấu trúc cây nhằm giảm thiểu bộ nhớ trong khai thác

HUIs. PB [14] là thuật toán dựa trên cơ sở cài đặt các chỉ mục và triển khai chiến lược cắt tỉa được thiết kế bởi Lan và cộng sự, nhằm đem đến hiệu quả cao trong khai thác HUIs.

2.4 Khai thác tập mẫu từ bộ dữ liệu không chắc chắn

Cơ sở dữ liệu không chắc chắn (uncertain database) là một loại cơ sở dữ liệu trong đó thông tin hoặc dữ liệu được lưu trữ không được xác định một cách chắc chắn hoặc đầy đủ. Cơ sở dữ liệu không chắc chắn xuất hiện khi chúng ta không chắc chắn về giá trị thực sự của một số liệu trong cơ sở dữ liệu. Điều này có thể phát sinh từ nhiều lý do: nhiễu, đo lường không chắc chắn hoặc không chắc chắn tự nhiên trong quá trình thu thập dữ liệu.

Trong các ứng dụng thực tế, dữ liệu của cơ sở dữ liệu không chắc chắn được thể hiện dưới dạng xác suất tồn tại và có giá trị nằm trong khoảng từ 0 đến 1.

Phương pháp khai thác tập mẫu từ bộ dữ liệu không chắc chắn thường phụ thuộc vào các kỹ thuật như lấy mẫu ngẫu nhiên hay khai thác tập mẫu không giám sát và bán giám sát. Liu [25] đã đề xuất một phương pháp để khai thác tập mẫu tuần tự không chắc chắn, sau đó Palacios và cộng sự [26] đã áp dụng phương pháp khai thác mẫu tuần tự không chắc chắn để theo dõi tình trạng động cơ máy bay với dữ liệu sức khỏe không chắc chắn.

CHƯƠNG 3. MÔ HÌNH TOÁN HỌC CỦA ĐỀ TÀI

Từ các kiến thức có được dựa trên các công trình liên quan đến khai phá k tập phổ biến có tiện ích cao đầu tiên trong cơ sở dữ liệu không chắc chắn, ta tiến hành phân tích đầu vào, đầu ra và các định nghĩa phục vụ cho bài toán.

Bảng 3.1 Danh sách giao dịch và bảng lợi nhuận của từng sản phẩm.

Giao dịch	Danh sách sản phẩm trong giao dịch	Giá trị tiện ích trong giao dịch
T_1	a: 0.9, c: 0.9, d: 0.6	a, c, d : 9 : 3, 1, 5
T_2	a: 0.9, b: 0.9, c: 0.7, d: 0.6, e: 0.4	a, b, c, d, e : 20 : 6, 2, 3, 5, 4
T_3	b: 0.5, c: 0.8, d: 0.9, f: 0.2	b, c, d, f : 17 : 2, 6, 5, 4
T_4	c: 0.9, e: 0.1, f: 0.5	c, e, f : 15 : 5, 4, 6
T_5	a: 0.4, b: 0.5, c: 0.9, d: 0.3	a, b, c, d : 23 : 6, 4, 3, 10
T_6	d: 0.9, e: 0.1, f: 0.6	d, e, f : 13 : 5, 4, 4
T_7	a: 0.9, b: 0.7	a, b: 5 : 3, 2

Mô hình toán học của bài toán được phát biểu như sau:

Cho:

- **i** = $\{i_1, i_2, ..., i_m\}$ là một tập hợp hữu hạn của m sản phẩm khác nhau.
- **D** = $\{T_1, T_2, ..., T_n\}$ là một tập các giao dịch, mỗi giao dịch bao gồm:
  - Danh sách các sản phẩm thuộc tập hợp **i** trong giao dịch.
  - Xác suất tồn tại của mỗi sản phẩm trong giao dịch, $pr(\mathbf{i}, T_q) \in \mathbf{D}$ là xác suất của item i trong giao dịch T_q $(1 \leq q \leq n)$ thuộc cơ sở dữ liệu **D**, có giá trị trong khoảng từ 0 đến 1 $(0 < pr(\mathbf{i}, T_q) < 1)$.
  - Tiện ích của mỗi sản phẩm giao dịch, $u(\mathbf{i}, T_q) \in \mathbf{D}$ là tiện ích của item i trong giao dịch T_q $(1 \leq q \leq n)$ thuộc cơ sở dữ liệu **D**.
- k là số lượng tập phổ biến có tiện ích cao đầu tiên theo tùy chọn của người dùng.
- δ là ngưỡng tiện ích tối thiểu theo tùy chọn của người dùng.

Kết quả của bài toán: tìm ra k tập phổ biến có tiện ích cao đầu tiên dựa theo ngưỡng tiện ích δ người dùng đặt ra từ bộ cơ sở dữ liệu không chắc chắn **D**.

Bảng 3.1 là ví dụ minh họa cho bộ cơ sở dữ liệu không chắc chắn **D**.

Các định nghĩa được sử dụng cho bài toán:

Định nghĩa 1: Xác suất mong đợi (expected probabitily) của mẫu $\boldsymbol{x}$ trong một giao dịch T_q, ký hiệu là pr($\boldsymbol{x}$, T_q), được định nghĩa là:

$$pr(\boldsymbol{x}, T_q) = \prod_{i \in \boldsymbol{x}} pr(\boldsymbol{i}, T_q)$$

Ví dụ: dựa theo Bảng 3.1, xác suất mong đợi của mẫu $\boldsymbol{x}$ = {a, b} trong giao dịch T_2 được tính tương ứng:

$$pr(\boldsymbol{x}, T_2) = \prod_{i \in \boldsymbol{x}} pr(\boldsymbol{i}, T_2)$$

$$= pr(a, T_2) * pr(b, T_2)$$

$$= 0.9 * 0.9 = 0.81$$

Định nghĩa 2: Độ hỗ trợ mong đợi (expected support) của một mẫu $\boldsymbol{x}$ trong bộ dữ liệu **D**, ký hiệu là expSup($\boldsymbol{x}$), được định nghĩa là:

$$expSup(\boldsymbol{x}) = \sum_{T_q \in \boldsymbol{D_x}} pr(\boldsymbol{x}, T_q)$$

Trong đó: $\mathbf{D}_x$ là tập các giao dịch có chứa mẫu $\boldsymbol{x}$ trong cơ sở dữ liệu **D**.

Ví dụ: dựa theo Bảng 3.1, ta có:

expSup(a) = 0.9 + 0.9 + 0.4 + 0.9 = 3.1

expSup(ab) = (0.9*0.9) + (0.4*0.5) + (0.9*0.7) = 1.64

Định nghĩa 3: Tiện ích của một mẫu $\boldsymbol{x}$ trong giao dịch T_q, ký hiệu là u($\boldsymbol{x}$, T_q) và được định nghĩa là:

$$u(\boldsymbol{x}, T_q) = \sum_{i \in \boldsymbol{x}} u(\boldsymbol{i}, T_q)$$

Ví dụ: dựa theo Bảng 3.1, tiện ích của mẫu $\boldsymbol{x}$ = {a, c} trong giao dịch T_1 được tính là:

$$u(\boldsymbol{x}, T_1) = u(a, T_1) + u(c, T_1) = 3 + 1 = 4$$

Định nghĩa 4: Tiện ích của một mẫu $\boldsymbol{x}$ trong cơ sở dữ liệu **D**, ký hiệu là u($\boldsymbol{x}$) và được định nghĩa là:

$$u(\boldsymbol{x}) = \sum_{T_q \in \boldsymbol{D_x}} u(\boldsymbol{x}, T_q)$$

Trong đó: D_x là tập các giao dịch có chứa mẫu $\boldsymbol{x}$ trong cơ sở dữ liệu **D**.

Ví dụ: Dựa theo Bảng 3.1, tiện ích của mẫu $\boldsymbol{x}$ = {a, c} được tính bằng:

$$u(\boldsymbol{x}) = u(\boldsymbol{x}, T_1) + u(\boldsymbol{x}, T_2) + u(\boldsymbol{x}, T_5)$$

$$= (3 + 1) + (6 + 3) + (6 + 3) = 22$$

Định nghĩa 5: Tiện ích của một giao dịch T_q được ký hiệu là tu(T_q), và được định nghĩa là:

$$tu(T_q) = \sum_{i \in Tq} u(\boldsymbol{i}, T_q)$$

Ví dụ: dựa theo Bảng 3.1, ta có:

tu(T_1) = u(a, T_1) + u(c, T_1) + u(d, T_1)

= 3 + 1 + 5 = 9

Định nghĩa 6: Tổng tiện ích của cơ sở dữ liệu **D**, ký hiệu là du, được định nghĩa như sau:

$$du = \sum_{T_q \in \boldsymbol{D}} tu(T_q)$$

Ví dụ: Tổng tiện ích của cơ sở dữ liệu **D** theo Bảng 3.1 được tính bằng:

du = 9 + 20 + 17 + 15 + 23 + 13 + 5 = 102

Định nghĩa 7: Mức tiện ích trọng số của một mẫu $\boldsymbol{x}$ là tổng các tiện ích của giao dịch mà chứa $\boldsymbol{x}$, được kí hiệu là twu($\boldsymbol{x}$) và có định nghĩa như sau:

$$twu(\boldsymbol{x}) = \sum_{\boldsymbol{x} \subseteq T_q \wedge T_q \in \boldsymbol{D}} tu(T_q)$$

Định nghĩa 8: Một mẫu $\boldsymbol{x}$ trong cơ sở dữ liệu **D** là một mẫu tiện ích cao, ký hiệu là HUP nếu và chỉ nếu tiện ích của nó cao hơn ngưỡng tiện ích tối thiểu và được tính là:

$$HUP \leftarrow \{\boldsymbol{x} | u(\boldsymbol{x}) \geq du \times \delta\}$$

Ví dụ: dựa theo Bảng 3.1, với ngưỡng tối thiểu δ bằng 20%, du = 102, xét tiện ích của tập mẫu {a} và tập mẫu {a, c}, ta thấy:

Tập mẫu {a} không phải là một HUP vì u(a) = 18 < 102 x 20% = 20,4

Tập mẫu {a, c} là một HUP vì u(a) = 22 > 102 x 20% = 20,4

Định nghĩa 9: Top-k tập mẫu phổ biến không chắc chắn có tiện ích cao trong cơ sở dữ liệu D, ký hiệu là TUHUFP, gồm các tập mẫu $\boldsymbol{x}$ thuộc cơ sở dữ liệu D và được định nghĩa là:

$$TUHUFP \leftarrow \{\boldsymbol{x} | \boldsymbol{x} \in HUP \cap expSup(\boldsymbol{x}) \geq expSup_{min}(TUHUFP)\}$$

Trong đó: $expSup_{min}(TUHUFP)$ là độ hỗ trợ mong đợi nhỏ nhất trong tập TUHUFP và được tính bằng:

$$expSup_{min}(TUHUFP) = \min(expSup(\boldsymbol{x}) | \boldsymbol{x} \in TUHUFP)$$

Định nghĩa 10: Giá trị tiện ích đường dẫn của đường dẫn p_j trong một cơ sở mẫu có điều kiện (conditional pattern base) của mẫu **x** ({**x**}-CPB) bằng với giá trị của tiện ích của node.

Ví dụ: giá trị tiện ích đường dẫn của đường dẫn {c, d} trong {d}-CPB là 6

Định nghĩa 11: Giá trị tiện ích đường dẫn của một mẫu trong đường đẫn trong một cơ sở mẫu có điều kiện bằng với giá trị tiện ích đường dẫn của đường dẫn.

Ví dụ: giá trị đường dẫn của mẫu c của đường dẫn {c, d} trong {d}-CPB là 6

Định nghĩa 12: Giá trị tiện ích đường dẫn của một mẫu trong một cơ sở mẫu có điều kiện là tổng giá trị tiện ích đường dẫn của mẫu đó trong đường đẫn.

Ví dụ: Giá trị đường dẫ tiện ích của mẫu {c} trong {d}-CPB là 6+8+11+13=38

Từ các định nghĩa trên, ta có cơ sở dữ liệu **D** không chắc chắn, bảng giá trị tiện ích, xác suất tồn tại của các tập mẫu, một ngưỡng tiện ích tối thiểu theo mong muốn của người dùng và độ hỗ trợ mong đợi thấp nhất trong tập TUHUFP. Vấn đề được hiểu ở đây là các mẫu sẽ được lưu vào trong TUHUFP khi mẫu đó thuộc HUP (giá trị tiện ích không nhỏ hơn du x δ) và có độ hỗ trợ mong đợi phải lớn hơn ngưỡng phổ biến (độ hỗ trợ mong đợi thấp nhất trong tập TUHUFP).

CHƯƠNG 4. PHƯƠNG PHÁP THỰC HIỆN

Có rất nhiều phương pháp đã được tìm ra nhằm khai thác các tập mẫu không chắc chắn cũng như khai thác các tập mẫu có tiện ích cao. Trong báo cáo này, chúng tôi sẽ sử dụng hai thuật toán là TUHUFP và TUHUFP_Growth để giải quyết bài toán khai thác k tập phổ biến có tiện ích cao đầu tiên trong bộ dữ liệu không chắc chắn.

4.1 Thuật toán TUHUFP

Thuật toán TUHUFP (Top Uncertain High Utility Frequent Pattern) sử dụng cấu trúc dữ liệu CUP-list, thông qua phương thức TUHUFP_Search cùng các chiến lược nhằm nâng cao hiệu quả khai thác để tìm ra tập phổ biến tiện ích cao đầu tiên từ bộ cơ sở dữ liệu không chắc chắn.

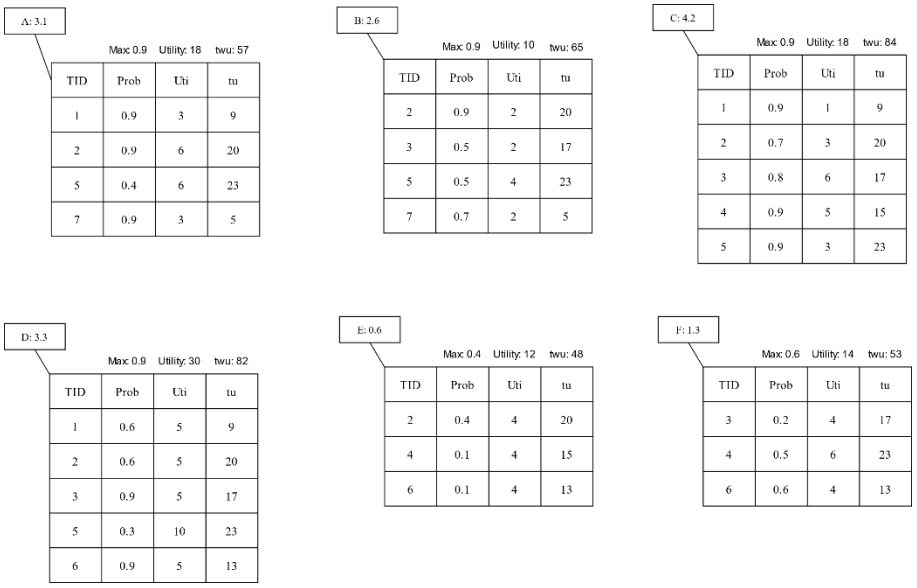
4.1.1 CUP-lists

CUP-list là cấu trúc dữ liệu được sử dụng để phục vụ khai thác top-k tập mẫu không chắc chắn có tiện ích cao (TUHUFP). Cấu trúc CUP-list gồm có:

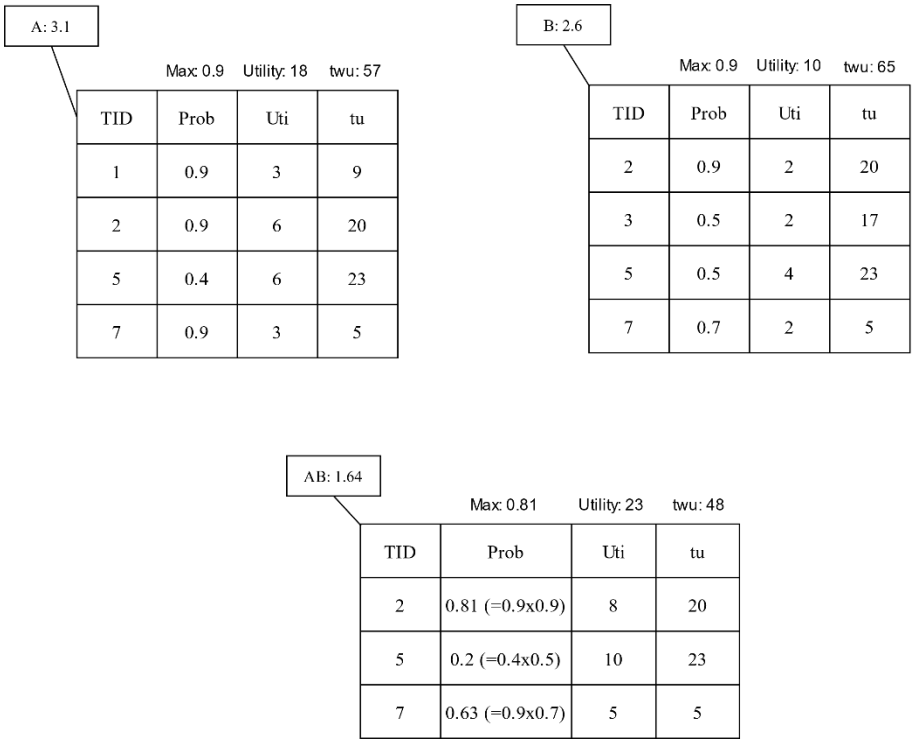
- Tên tập mẫu (pattern name)
- Độ hỗ trợ mong đợi (expected support) của mẫu
- Một tập hợp chứa dữ liệu của mẫu gọi là TEP-list
- Thuộc tính max để lưu giá trị xác suất mong đợi lớn nhất của mẫu trong TEP-List. Ngưỡng tối đa (overestimate) của một tập XY được xác định bằng expSup(X) x max(Y). Bằng việc tính toán trước ngưỡng tối đa của một tập mẫu và so sánh với ngưỡng đặt ra trong top-k, chúng ta có thể tính toán các tập kết hợp có tồn tại top-k hay không, từ đó cắt giảm không gian lưu trữ và nâng cao hiệu suất khai thác [24].
- Thuộc tính utility để lưu giá trị tiện ích của mẫu trong cơ sở dữ liệu.
- Thuộc tính TWU (transaction-weight-utility) là tổng các tiện ích của giao dịch trong TEP-List.
- Dữ liệu của TEP-list gồm:
    - Địa chỉ giao dịch (TID)

- Xác suất mong đợi (expected probability) của mẫu trong mỗi giao dịch
- Tiện ích của mẫu trong mỗi giao dịch
- Tiện ích của mỗi giao dịch

Dựa theo dữ liệu ví dụ ở Bảng 3.1, Hình 4.1 mô tả cấu trúc CUP-list của các sản phẩm có trong cơ sở dữ liệu. Bằng việc lưu trữ dữ liệu các mẫu theo cấu trúc này, chúng ta có thể tính toán độ hỗ trợ mong đợi và tiện ích của mẫu một cách hiệu quả mà không phải duyệt lại cơ sở dữ liệu.



Hình 4.1 Mô hình CUP-list



Hình 4.2 Mô hình liên kết CUP-list cho mẫu AB

Hình 4.2 thể hiện quá trình kết hợp mẫu {A, B} từ CUP-list item A và CUP-list item B. Đầu tiên, ta xác định TEP-list của mẫu từ các TID trùng lặp (2, 5, 7). Dựa vào các định nghĩa 1 và 3 để xác định các giá trị xác suất mong đợi và tiện ích của mẫu trong TEP-list.

4.1.2 Các chiến lược nâng cao hiệu quả khai thác

Chiến lược 1 – nâng ngưỡng: Danh sách k tập phổ biến có tiện ích cao đầu tiên trong cơ sở dữ liệu được lưu trong một mảng sắp xếp giảm dần độ hỗ trợ mong đợi (expSup). Từ đó độ hỗ trợ mong đợi của phần tử cuối cùng trong mảng chính là ngưỡng của top-k. Khi có một tập mẫu có độ hỗ trợ cao hơn độ hỗ trợ của phần tử cuối mảng, tập mẫu mới sẽ được thêm vào top-k và xóa đi phần tử cuối, đồng thời cập nhật ngưỡng hỗ trợ mong đợi theo phần tử cuối hiện tại. Chiến lược này được áp dụng để giảm thiểu thời gian tìm kiếm và thay thế tập mẫu trong danh sách top-k.

Chiến lược 2 – cắt tỉa theo ngưỡng được nâng: trong quá trình tạo các tập mẫu mới trong phương thức **TUHUFP_Search**, thuật toán sẽ không tạo ra tập mẫu

mới nếu độ hỗ trợ mong đợi của mẫu tham gia thấp hơn ngưỡng hỗ trợ mong đợi. Chiến lược này sẽ giúp giảm thiểu đáng kể thời gian xử lý và không gian lưu trữ của thuật toán.

Chiến lược 3 – cắt tỉa theo TWU: trong quá trình khai thác các tập mẫu, thuật toán sẽ có một danh sách chứa các tập mẫu gọi là danh sách ứng viên sẽ tham gia vào thuật toán. Thuật toán sẽ không thêm tập mẫu mới vừa được tạo vào danh sách ứng viên nếu TWU của tập mẫu đó nhỏ hơn ngưỡng tiện ích. Điều này giúp giảm số lượng ứng viên tham gia một cách đáng kể.

Chiến lược 4 – ngưỡng tối đa: Ngưỡng tối đa (overestimate) của một tập XY được xác định bằng expSup(X) x max(Y). Nếu ngưỡng tối đa nhỏ hơn ngưỡng đặt ra trong top-k thì ta sẽ bỏ qua tập mẫu này, ngược lại thì tập mẫu này sẽ trở thành tập mẫu "tiềm năng" trong top-k, từ đó cắt giảm không gian lưu trữ và nâng cao hiệu suất khai thác [16].

4.1.3 Mã giả thuật toán TUHUFP

Bước quan trọng trong thuật toán đề xuất được gọi là TUHUFP và được mô tả như sau:

1. Tìm kiếm CUP-list của tập 1-mẫu (chứa 1 phần tử) trong cơ sở dữ liệu **D** và sắp xếp theo thứ tự giảm dần độ hỗ trợ mong đợi.

2. Lấy k (hoặc nhỏ hơn tùy vào số lượng phần tử trong cơ sở đữ liệu) phần tử là HUP đầu tiên vào danh sách top-k.

3. Chọn lọc các mẫu tham gia thông qua TWU. Nếu TWU của mẫu đó nhỏ hơn ngưỡng tiện ích thì sẽ bị bỏ qua, ngược lại sẽ tham gia vào quá trình tìm kiếm cảu thuật toán.

4. Thiết lập threshold là độ mong đợi tin cây của phần tử thấp nhất trong top-k khi top-k có đủ k phần tử.

5. Gọi phương thức **TUHUFP_Search** truyền vào giá trị k và danh sách mẫu tham gia. TUHUFP_Search sử dụng chiến lược chia để trị để tạo ra các CUP-list của tập mẫu lớn hơn từ các CUP-list của tập mẫu tham gia. Với mỗi tập mẫu, nếu độ hỗ trợ mong đợi thấp hơn ngưỡng hỗ trợ mong đợi hoặc tiện ích của tập mẫu thấp hơn

ngưỡng tiện ích thì sẽ bỏ qua. Nếu không, thuật toán sẽ thêm tập mẫu vào kết quả và kết hợp chúng để tìm kiếm các tập mẫu lớn hơn. Chiến lược này sẽ tìm kiếm đến khi tất cả mẫu được xem xét.

THUẬT TOÁN 1: Thuật toán TUHUFP
Đầu vào: U: một cơ sở dữ liệu không chắc chắn, H: cơ sở dữ liệu tiện ích, percentage: ngưỡng tiện ích mong muốn của người dùng (%), k: số lượng UHUFP cần tìm. **Đầu ra:** k tập mẫu phổ biến không chắc chắn có tiện ích cao đầu tiên
1. Đọc cơ sở dữ liệu U và H để tạo danh sách CUP (1-mẫu) 2. Tính tiện ích của cơ sở dữ liệu: *databaseUtil* ← *SUM(tu(Tq)* ∈ *H)* 3. *minUtil* ← *databaseUtil*percentage* 4. *threshold* ← *0* 5. Sắp xếp lại danh sách CUP (1-mẫu) theo *expSup* 5. *Candidates* ← *Candidates* ∪ *CUPx* ∈ *CUP, CUPx.TWU* ≥ *minUtil* 6. *topUHUFP* ← *topUHUFP* ∪ *CUPx* ∈ *CUP, CUPx.utility* ≥ *minUtil* 7. *threshold* ← phần tử cuối *CUPx* ∈ *CUP, topUHUFP.size = k* 8. *TUHUFPSearch(Candidates, threshold, minUtil, k)*

4.2 Thuật toán TUHUFP-Growth

Thuật toán TUHUFP-Growth (Top Uncertain High Utility Frequent Pattern Growth) sử dụng cấu trúc dữ liệu UHUFP-tree, một phiên bản mở rộng của FP-tree để lưu trữ và khai thác các mẫu phổ biến không chắc chắn có tiện ích cao từ cơ sở dữ liệu không chắc chắn. TUHUFP-Growth được xây dựng để tối ưu hóa quá trình khai thác bằng cách sử dụng các chiến lược cắt tỉa và cấu trúc dữ liệu hiệu quả.

4.2.1 UHUFP-Tree

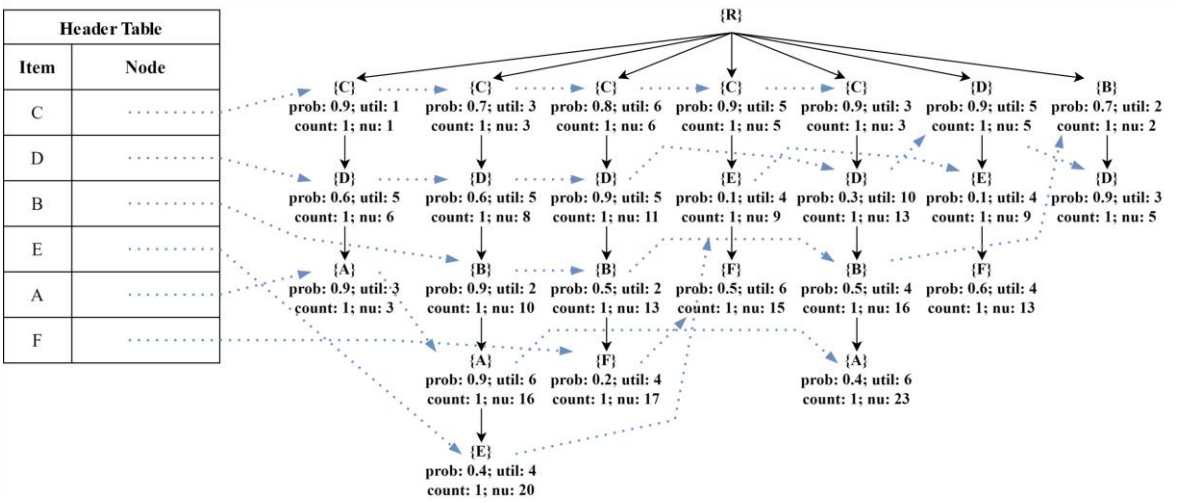
UHUFP-tree là một cây dạng FP-tree mở rộng, có thể lưu trữ thông tin về các mẫu phổ biến và tiện ích của chúng trong cơ sở dữ liệu không chắc chắn. Cấu trúc UHUFP-tree bao gồm:

- Gốc (root): gốc của cây UHUFP-tree.
- Bảng header (header_table): một danh sách chứa các sản phẩm sắp xếp giảm dần theo độ hỗ trợ mong đợi cùng các node của item đó trong cây

Mỗi node trong cây có các thuộc tính:

- Tên mẫu
- Xác suất mong đợi của mẫu
- Tiện ích của mẫu
- Bộ đếm (count) số lần xuất hiện của node trong cây
- Nu là tiện ích của node (giá trị tiện ích ước tính)

Cấu trúc UHUFP-tree cho phép việc lưu trữ các thông tin cần thiết về tiện ích và độ hỗ trợ mong đợi của các mẫu, giúp giảm thiểu yêu cầu bộ nhớ và tăng tốc quá trình khai thác.



Hình 4.3 Mô hình UHUFP-tree

4.2.2 Các chiến lược nâng ngưỡng hiệu quả

Chiến lược khai thác trong UHUFP-tree tập trung vào việc tối ưu hóa quá trình tìm kiếm các mẫu phổ biến có tiện ích cao thông qua các chiến lược cắt tỉa tương tự như trong TUHUFP:

Chiến lược 5 – loại bỏ các mục không hứa hẹn toàn cục: Trong chiến lược này, các mục không hứa hẹn và tiện ích của chúng được loại bỏ khỏi các tiện ích giao dịch trong quá trình xây dựng cây UHUFP toàn cục. Nguyên tắc của chiến lược là loại bỏ thông tin của các mục không hứa hẹn ra khỏi cơ sở dữ liệu vì các mục này không đóng vai trò quan trọng trong việc hình thành các tập mục có tiện ích cao. Chỉ các tập con mở rộng của các mục hứa hẹn mới có khả năng trở thành các tập mục có tiện ích cao.

Chiến lược 6 – loại bỏ tiện ích nút toàn cục: Đối với bất kỳ nút nào trong cây UHUFP toàn cục, tiện ích của các nút con của nó sẽ bị loại bỏ khỏi tiện ích của nút trong quá trình xây dựng cây UHUFP toàn cục. Giả sử $<i_1, i_2, \ldots, i_n>$ là danh sách các mục hứa hẹn được sắp xếp theo thứ tự giảm dần của các giá trị TWU trong bảng tiêu đề toàn cục. Vì các mục i_{k+1}, i_{k+2}, ..., in không liên quan đến i_k-CPB và $i_{k\text{-Tree}}$, chúng sẽ không được chứa trong bất kỳ PHUI (Potential High Utility Itemset) nào của i_k-Tree. Do đó, các tiện ích của chúng có thể bị loại bỏ khỏi nút ik trong cây UP toàn cục.

Chiến lược 7 – loại bỏ các mục không hứa hẹn cục bộ: Các giá trị tiện ích tối thiểu của các mục không hứa hẹn sẽ bị loại bỏ khỏi tiện ích đường dẫn trong quá trình xây dựng cây UHUFP cục bộ. Dựa trên nguyên lý của chiến lược, trong cây mẫu điều kiện, các mục không hứa hẹn cục bộ và tiện ích của chúng có thể bị loại bỏ. Vì tiện ích mục tối thiểu của một mục không hứa hẹn cục bộ trong một đường dẫn luôn bằng hoặc nhỏ hơn tiện ích thực của nó trong đường dẫn, chúng ta có thể loại bỏ tiện ích mục tối thiểu này khỏi các đường dẫn của cây mẫu điều kiện mà không làm mất bất kỳ PHUI (Potential High Utility Itemset) nào.

Chiến lược 8 – giảm tiện ích nút cục bộ: Các giá trị tiện ích mục tối thiểu của các nút con được giảm đi trong quá trình xây dựng cây UP cục bộ. Giả sử có một

danh sách các mục hứa hẹn <i_1', i_2', …, i_n'> được sắp xếp theo thứ tự giảm dần của các giá trị tiện ích đường dẫn cục bộ trong bảng tiêu đề cục bộ. Dựa trên nguyên lý của chiến lược DGN, các mục i_{k+1}', i_{k+2}', ..., in' và tiện ích của chúng có thể bị loại bỏ khỏi i_k'-Tree. Vì tiện ích mục tối thiểu của một mục i_m' ($k+1 \leq m \leq n$) trong một đường dẫn luôn bằng hoặc nhỏ hơn tiện ích thực của nó trong đường dẫn, chúng ta cũng có thể loại bỏ tiện ích mục tối thiểu của i_m' khỏi các đường dẫn của i_k'-Tree mà không làm mất bất kỳ PHUI (Potential High Utility Itemset) nào.

4.2.3 Mã giả thuật toán TUHUFP-Growth

Thuật toán TUHUFP-Growth sử dụng cấu trúc cây UHUFP để tìm ra các tập mẫu phổ biến có tiện ích cao đầu tiên được mô tả như sau:

1. Từ hai cơ sở dữ liệu ta có được các mẫu cùng với TWU tương ứng và xây dựng cây UHUFP toàn cục.
2. Bảng header của cây sẽ được sắp xếp theo thứ tự giảm dần của độ hỗ trợ mong đợi của các mẫu và bảng header này chỉ chứa các mẫu có TWU lớn hơn hoặc bằng với ngưỡng tiện ích.
3. Ta chỉ lấy k phần tử đầu tiên trong header để tiến hành khai thác các mẫu mới.
4. Thiết lập threshold và ngưỡng tiện ích (threshold ban đầu sẽ bằng 0 và ngưỡng tiện ích minUtil là percentage nhân với giá trị tiện ích của cơ sở dữ liệu).
5. Gọi phương thức **TUHUFP_Growth** với các tham số như cây UHUFP toàn cục, threshold, ngưỡng tiện ích và k là số mẫu cần tìm. Phương thức này sẽ sử dụng phương thức chia để trị nhằm chia nhỏ cây UHUFP toàn cục thành các cây UHUFP nhỏ hơn của từng mẫu (cây cục bộ) cho việc khai thác các tập mẫu. Với mỗi tập mẫu, nếu độ hỗ trợ mong đợi thấp hơn ngưỡng hỗ trợ mong đợi hoặc tiện ích của tập mẫu thấp hơn ngưỡng tiện ích thì sẽ bỏ qua. Nếu không, thuật toán sẽ thêm tập mẫu vào kết quả và kết hợp chúng để tìm kiếm các tập mẫu lớn hơn. Chiến lược này sẽ tìm kiếm đến khi tất cả mẫu được xem xét.

THUẬT TOÁN 2: Thuật toán TUHUFP_Growth

Đầu vào: U: một cơ sở dữ liệu không chắc chắn, H: cơ sở dữ liệu tiện ích, percentage: ngưỡng tiện ích mong muốn của người dùng (%), k: số lượng UHUFP cần tìm.

Đầu ra: k tập mẫu phổ biến không chắc chắn có tiện ích cao đầu tiên

1. Đọc cơ sở dữ liệu U và H để tính được TWU của các item
2. Tính tiện ích của cơ sở dữ liệu: *databaseUtil* ← *SUM(tu(Tq)* ∈ *H)*
3. *minUtil* ← *databaseUtil*percentage*
4. *threshold* ← *0*
5. Đọc cơ sở dữ liệu lần hai để tạo cây UHUFP
6. Sắp xếp lại các item trong header table theo *expSup* (giảm dần)
7. *New_header_table* ← *New_header_table* ∪ item ∈ *header_table[0:k], item.TWU* ≥ *minUtil*
8. *topUHUFP* ← ∅
9. *TUHUFP_Growth(UHUFP_tree, threshold, minUtil, k, prefix)*

CHƯƠNG 5. TRIỂN KHAI GIẢI PHÁP

5.1 Xử lý dữ liệu

Để triển khai khai thác k tập mẫu tiện ích cao phổ biến đầu tiên, đề tài yêu cầu hai bộ dữ liệu:

- Bộ dữ liệu không chắc chắn: gồm danh sách các giao dịch và danh sách các sản phẩm trong mỗi giao dịch, ứng với mỗi sản phẩm trong giao dịch chứa xác suất tồn tại (existential probability) của sản phẩm.
- Bộ dữ liệu tiện ích: gồm danh sách các sản phẩm, tiện ích tương ứng của sản phẩm trong các giao dịch và tiện ích của mỗi giao dịch.

Các bộ dữ liệu trên được tham khảo từ thư viện nguồn dữ liệu mở SPMF. Trong đó, bộ dữ liệu không chắc chắn sẽ được bổ sung thêm xác suất tồn tại của sản phẩm bằng phương pháp sinh ngẫu nhiên để phù hợp với yêu cầu của đề tài.

5.2 Sơ đồ lớp chứa thuật toán TUHUFP

Giải pháp được triển khai với các lớp sau:

- Tep: lớp này sử dụng để tạo đối tượng Tep bên trong TEP-List của CUP; bao gồm các thuộc tính như TID, xác suất tồn tại, giá trị tiện ích, và giá trị tiện ích của giao dịch tại TID đó, các phương thức cơ bản như khởi tạo, các getters và setters.
- Cup: lớp được sử dụng để tạo nên CUP tham gia vào quá trình khai thác của thuật toán, các thuộc tính bao gồm: tên, độ hỗ trợ mong đợi, giá trị tiện ích, TEP-List, giá trị max trong TEP-List, tổng các tiện ích giao dịch trong TEP-List; cùng với các phương thức khởi tạo, setters, getters.
- AlgorithmTUHUFP: lớp này chứa thuật toán TUHUFP để khai thác k tập mẫu tiện ích cao phổ biến đầu tiên. Các thuộc tính trong lớp này bao gồm: danh sách kết quả, một Map chứa các 1-mẫu tham gia, ngưỡng tiện ích và ngưỡng phổ biến.

Các phương thức của lớp AlgorithmTUHUFP:

+ readData(filePath: String, percentage: double, k: int): đọc file chứa các cơ sở dữ liệu để chuẩn bị cho quá trình xử lý.

+ processData(itemName: String, probLine: String, utilLine: String, TID: Integer): xử lý dữ liệu được truyền từ readData() và tạo ra CUP-list 1-mẫu.

+ getFitstUHUFP(minUtil: T2, k: int): phương thức này trả về một danh sách các mẫu tham gia vào thuật toán, đồng thời sẽ thêm các mẫu thỏa điều kiện về ngưỡng lợi ích vào kết quả với độ dài k.

+ TUHUFPSearch(currentCup: List<Cup>, k: int): phương thức chính của thuật toán, thực hiện khai thác các tập mẫu và thêm vào kết quả cuối cùng khi thỏa các ngưỡng và trả về k tập mẫu không chắc chắn tiện ích cao phổ biến đầu tiên.

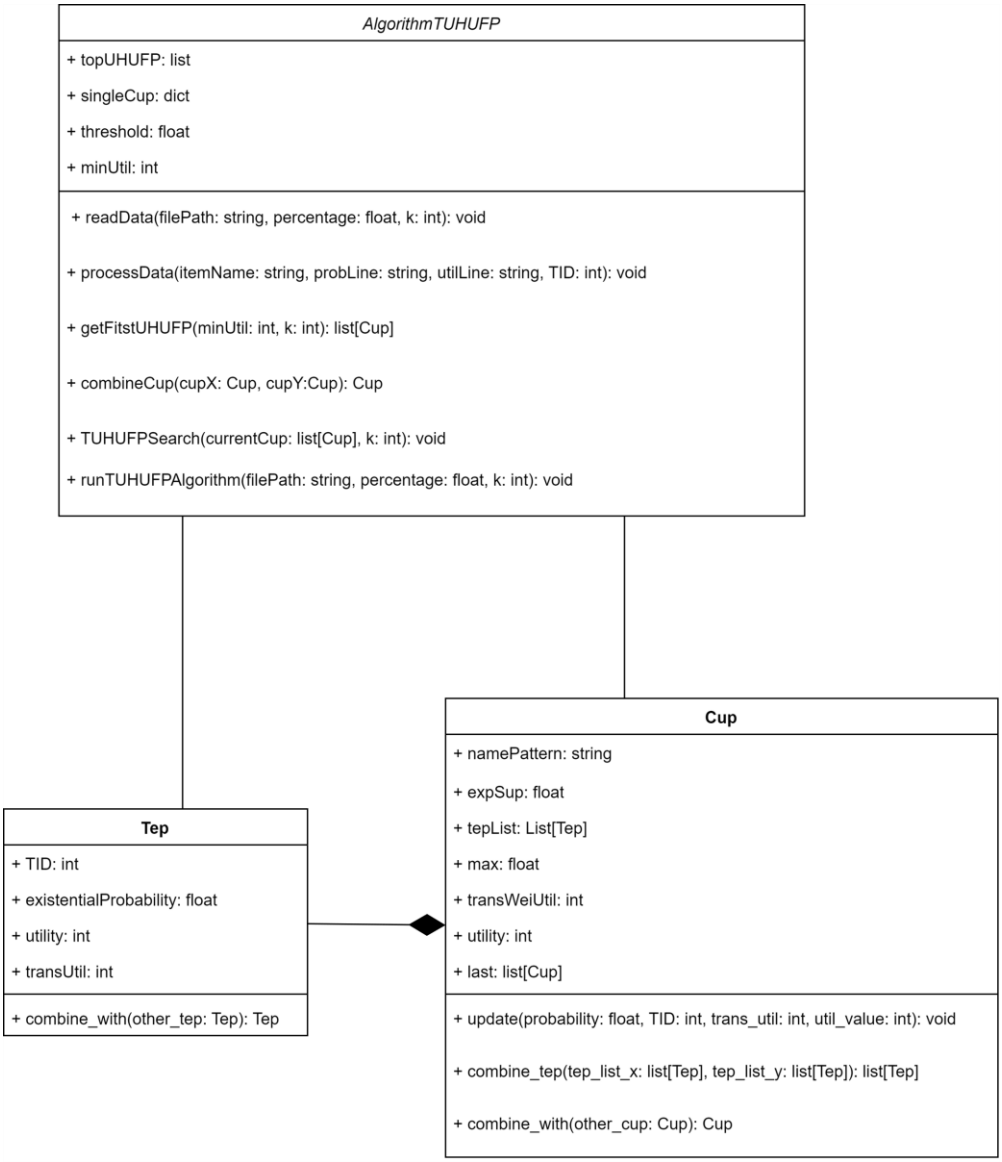
+ runTUHUFPAlgorithm(filePath: String, percentage: double, k: int): phương thức được dùng để chạy thuật toán với các cài đặt cần thiết.

Các phương thức của lớp Cup

+ update(probability: float, TID: int, trans_util: int, util_value: void): hàm cập nhật lại thông tin của CUP khi thêm một transaction mới.

+ combine_tep(tep_list_x: List<Tep>, tep_list_y: List<Tep>): phương thức sử dụng để ghép 2 TEP-List của 2 CUP lại với nhau và trả về một TEP-List mới.

+ combine_with(otherCup: Cup): phương thức sử dụng để ghép 2 CUP lại với nhau và trả về một CUP mới.



Hình 5.1 Sơ đồ lớp chứa thuật toán TUHUFP

5.3 Thiết lập thuật toán TUHUFP

Sau khi đã có được danh sách CUP-lists 1-mẫu, bằng cách sắp xếp danh sách này theo thứ tự giảm dần của độ hỗ trợ mong đợi (expSup), đưa ra k số lượng tập mẫu đầu tiên là tập mẫu có tiện ích cao vào danh sách kết quả (số lượng mẫu được đưa vào kết quả có thể nhỏ hơn k). Đồng thời, kiểm tra 1-mẫu trong danh sách theo chiến lược 3 (TWU lớn hơn hay nhỏ hơn ngưỡng tiện ích tối thiểu) để có được một danh sách ứng viên ($\mathbf{u}_1$).

Nếu số lượng mẫu trong danh sách kết quả bằng k thì thiết lập threshold bằng độ hỗ trợ mong đợi của phần tử thấp nhất trong danh sách. Ngược lại, threshold sẽ được thiết lập cho đến khi số lượng mẫu đầu ra bằng với k.

TUHUFP_Search là phương thức sử dụng để khai thác các tập mẫu phổ biến không chắc chắn có tiện ích cao từ danh sách **u**.

Gọi phương thức **TUHUFP_Search** với chiến lược chia để trị. Đầu tiên với danh sách ứng viên k-mẫu, thuật toán sẽ kết hợp các phần tử trong danh sách với nhau để tạo ra (k+1)-mẫu. Nếu (k+1)-mẫu này có độ hỗ trợ mong đợi nhỏ hơn ngưỡng hỗ trợ mong đợi thì thuật toán sẽ bỏ qua. Ngược lại, thuật toán sẽ kiểm tra:

1) Nếu giá trị tiện ích của mẫu có lớn hơn ngưỡng tiện ích thì thêm vào danh sách kết quả;

2) Gọi $\mathbf{u}_{k+1}$ là danh sách ứng viên mới, nếu TWU của mẫu lớn hơn ngưỡng tiện ích thì thêm mẫu vào $\mathbf{u}_{k+1}$, tiếp tục gọi **TUHUFP_Search** với danh sách ứng viên để tạo danh sách (k+2)-mẫu. Chiến lược này được sử dụng cho đến khi tất cả các mẫu đã được xem xét.

Trong phương thức TUHUFP_Search sẽ có các chiến lược được nên ở 4.2:

1) Khi ta ghép mẫu đầu tiên với các mẫu còn lại trong danh sách để tạo ra một ứng viên mới thì ta cần phải kiểm tra xem ứng viên được tạo ra có phải là ứng viên “tiền năng phổ biến” hay không thông qua kiểm tra overestimate với threshold được nêu ở chiến lược 4. Nếu overestimate nhỏ hơn threshold thì sẽ bỏ qua tất cả phần tử còn lại trong danh sách.
2) Sau khi mẫu mới được tạo thành, nếu độ hỗ trợ mong đợi của mẫu lớn hơn threshold và là HUP thì thêm vào danh sách kết quả đồng thời đặt lại threshold, và tiếp tục kiểm tra nếu TWU của mẫu lớn hơn ngưỡng tiện ích, thì thêm vào danh sách ứng viên cho vòng tìm kiếm tiếp theo. Điều này được nêu ở chiến lược 1, 2 và 3.

THUẬT TOÁN 3: Phương thức TUHUFP_Search

Đầu vào: P: một danh sách chứa các Cup, threshold: ngưỡng phổ biến, minUtil: ngưỡng tiện ích mong muốn của người dùng, và k là số lượng UHUFP cần tìm.

Đầu ra: k tập mẫu phổ biến không chắc chắn có tiện ích cao đầu tiên

```
1.  Nếu P.size > 1 thì:
2.      Với mỗi Px ∈ P, thực hiện:
3.          ListofnewCUP ← ∅
4.          Với mỗi Py ∈ P, y > x thực hiện
5.              overestimate ← Px.expSup*Py.max
6.              // chiến lược 4
7.              Nếu overetimate < threshold , thì
8.                  Thực hiện Px tiếp theo, quay lại 2
9.              Kết thúc
10.             combined ← combineCup(Px, Py)
11.             // chiến lược 2
12.             Nếu combined.expSup > threshold, thì
13.                 // thêm vào kết quả khi combined là HUP
14.                  và đặt lại threshold khi cần
15.                 // chiến lược 1
16.                 TUHUFPSearchHelper(combined, k)
17.                 // chiến lược 3
18.                 Nếu combined.TWU ≥ minUtil, thì
```

19.	*ListofnewCUP ← ListofnewCUP ∪ combined*
20.	**Kết thúc**
21.	**Kết thúc**
22.	**Kết thúc**
24	*TUHUFP_Search(ListofnewCUP, threshold, minUtil, k)*
25.	**Kết thúc**
26.	**Kết thúc**

5.4 Sơ đồ lớp thuật toán TUHUFP_Growth

Giải pháp được triển khai với các lớp sau:

- Node: lớp này sử dụng để lưu thông tin của mẫu trong cây; bao gồm các thuộc tính như item, xác suất tồn tại, giá trị tiện ích của item, số lần xuất hiện và tiện ích của node.
- UHUFP_tree: lớp được sử dụng để tạo cây UHUFP từ bộ dữ liệu, các thuộc tính bao gồm: root là gốc của cây UHUFP có giá trị là null và header_table là một danh sách chứa các sản phẩm sắp xếp giảm dần theo độ hỗ trợ mong đợi cùng các node của item đó trong cây.
- TUHUFPGrowth: lớp này chứa thuật toán TUHUFP-Growth để khai thác k tập mẫu tiện ích cao phổ biến đầu tiên. Các thuộc tính trong lớp này bao gồm: danh sách kết quả, ngưỡng tiện ích và ngưỡng phổ biến.

Các phương thức của lớp UHUFP_tree:

+ insert_reorganized_transaction(transaction: dict, utilities: dict, sorted_items: dict): phương thức này chịu trách nhiệm thêm từng giao dịch vào cây UHUFP, tạo nên cây UHUFP toàn cục với các tham số transaction chứa xác suất tồn tại của các mẫu, utilities chứa giá trị tiện ích của các mẫu và sorted_item là danh sách giảm dần theo TWU của các mẫu.

+ insert_reorganized_path(transaction: dict, utilities: dict, sorted_items: dict, miu: dict): tương tự như insert_reorganized_transaction nhưng thêm từng path vào cây UHUFP để tạo cây cục bộ cho một mẫu nhất định, miu là danh sách chứa các mẫu trong path cùng với giá trị tiện ích nhỏ nhất trong path.

+ prune_patterns(k: int, item_twu: dict, minUtil: int): phương thức này nhằm sắp xếp header table theo thứ tự giảm dần của độ hỗ trợ mong đợi và chỉ lấy k mẫu trong header table.

+ prune_tree(node: Node): xóa đi các node mà không có trong danh sách header table mới sau khi sử dụng phương thức prune_patterns.

Các phương thức của lớp TUHUFPGrowth:

+ calculate_expected_support(paths: list, prefix: list): tính giá trị độ hỗ trợ mong đợi của prefix thông qua path.

+ calculate_utility(paths: list, prefix: list): tính giá trị tiện ích của prefix thông qua path.

+ TUHUFP_Growth(tree: UHUFPtree, k: int, minSup: float, minUtil: float, miut: dict, prefix: list, top_k: list): đây là phương thức quan trọng của thuật toán. Chịu trách nhiệm xây dựng cây UHUFP cục bộ và khai thác các tập mẫu phổ biến có giá trị tiện ích cao từ những cây đó.

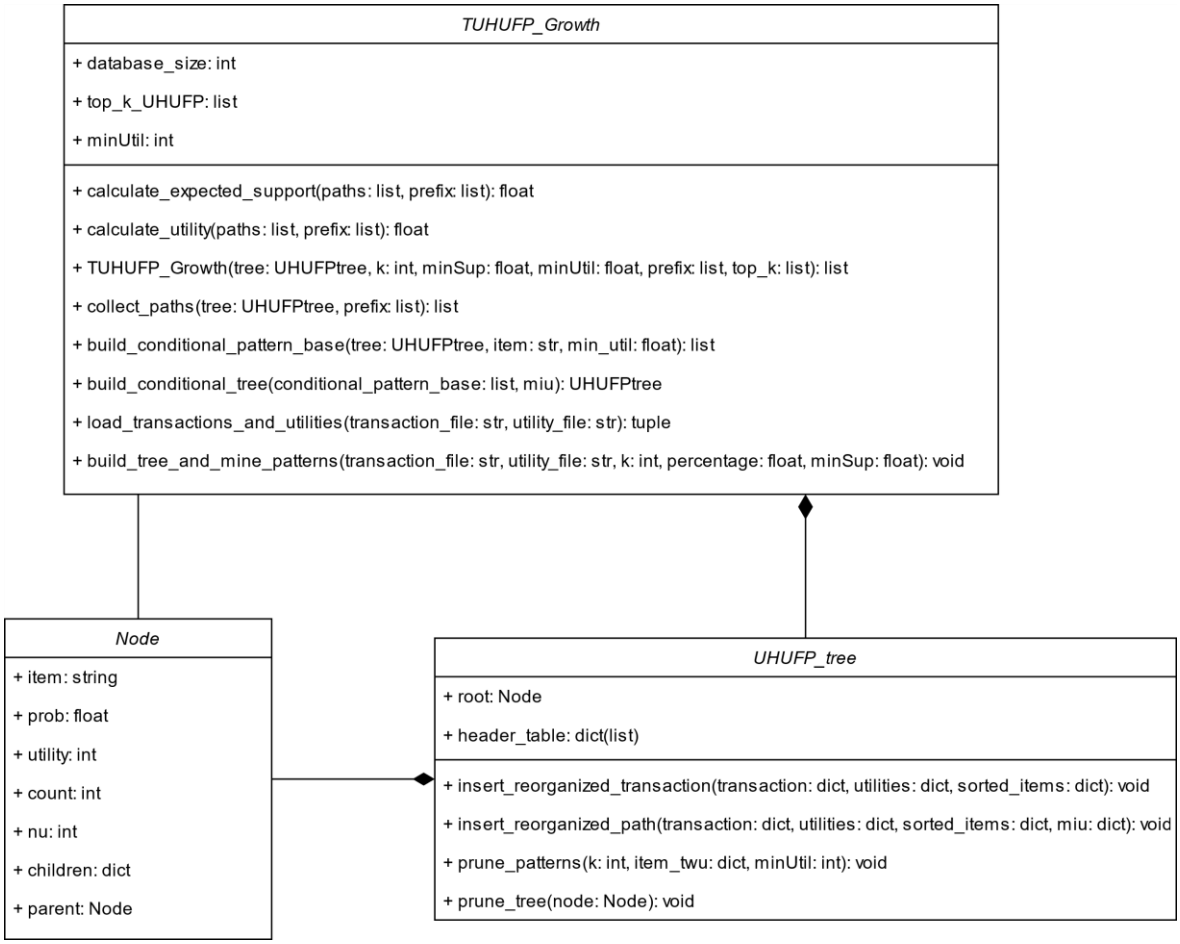
+ collect_paths(tree: UHUFPtree, prefix: list): thu thập path của prefix trong cây UHUFP.

+ build_conditional_pattern_base(tree: UHUFPtree, item: str, min_util: float): phương thức chịu trách nhiệm tổng hợp và thu thập các path cần thiết của item, các chiến lược giảm tiện ích được áp dụng tại phương thức này.

+ build_conditional_tree(conditional_pattern_base: list, miu: dict): xây dựng cây UHUFP cục bộ cho mẫu từ CPB (conditional pattern base)

+ load_transactions_and_utilities(transaction_file: str, utility_file: str): đọc các file cơ sở dữ liệu.

+ build_tree_and_mine_patterns(transaction_file: str, utility_file: str, k: int, percentage: float, minSup: float): phương thức để chạy thuật toán với các cài đặt cần thiết.



Hình 5.2 Sơ đồ lớp chứa thuật toán TUHUFP-Growth

5.5 Thiết lập thuật toán TUHUFP_Growth

Thuật toán TUHUFP_Growth sử dụng cây UHUFP để tiến hành khai thác các mẫu có giá trị tiện ích cao phổ biến đầu tiên. Thuật toán thực hiện đọc dữ liệu 2 lần, lần đầu tiên tính được TWU của từng mẫu từ mẫu dữ liệu và các mẫu không hứa hẹn sẽ bị loại bỏ khỏi các giao dịch. Với lần thứ hai, ta sẽ có được các giao dịch mới chỉ chứa các mẫu hứa hẹn (reorganized transactions) và thêm từng giao dịch và tạo nên cây UHUFP toàn cục (áp dụng chiến lược 5 và 6). Đồng thời các mẫu hứa hẹn sẽ

được liệt kê trong header table theo thứ tự giảm dần của độ hỗ trợ mong đợi. Sau khi hoàn thành cây UHUFP toàn cục ta tiến hành khai thác các mẫu UHUFP:

1. Duyệt từng mẫu trong header table lần lượt để tạo mẫu mới, kiểm tra overestimate của mẫu để chắc rằng đây là một mẫu tiềm năng (chiến lược 4). Tạo một mẫu mới từ mẫu trong header table với tiền tố (prefix) của mẫu (ban đầu sẽ được cho là rỗng), ta có được mẫu mới (new_prefix) kí hiệu ip.

2. Tiếp theo, thu thập các đường dẫn của mẫu mới vừa tạo được kí hiệu như sau {ip}-CPB (conditional pattern base của ip), áp dụng các chiến lược 7, 8 trong quá trình này. Ta có được giá trị tiện ích cục bộ của mẫu tại đường và nếu giá trị này lớn hơn hoặc bằng ngưỡng tiện ích thì đây được xem là một mẫu tiện ích cao tiềm năng:

- Thu thập các đường dẫn của mẫu cùng với giá trị tiện ích cục bộ của các mục trong đường dẫn. Chọn lọc được các mục hứa hẹn và loại bỏ các mục không hứa hẹn cùng giá trị tiện ích của chúng (giá trị nhỏ nhất trong đường dẫn) ra khỏi đường dẫn khi giá trị tiện ích cục bộ của mục đó nhỏ hơn ngưỡng tiện ích (chiến lược 7)
- Ta có được các RPs (reoganized paths) chứa các mục hứa hẹn và giá trị tiện ích của đường dẫn đã được điều chỉnh (theo chiến lược 7) (conditional pattern base - CPB). Thêm từng đường dẫn và tạo cây UHUFP cục bộ.

Sau đó, ta tính các giá trị cần thiết của mẫu (giá trị tiện ích, độ hỗ trợ mong đợi) từ CPB và thêm vào danh sách kết quả nếu thỏa điều kiện threshold.

3. Xây dựng cây UHUFP cục bộ dành cho mẫu mới để chuẩn bị cho quá trình khai thác tiếp theo. Áp dụng chiến lược 8, node.nu sẽ được điều chỉnh trong quá trình xây dựng cây cục bộ. Nếu cây UHUFP này tồn tại các node con thì thực hiện gọi thuật toán TUHUFP_Growth với cây vừa được tạo của mẫu mới. Quá trình này sẽ kết thúc khi các mẫu được duyệt qua hết.

THUẬT TOÁN 4: Phương thức TUHUFP_Growth

Đầu vào: tree: cây UHUFP, threshold: ngưỡng phổ biến, minUtil: ngưỡng tiện ích mong muốn của người dùng, và k là số lượng UHUFP cần tìm, prefix: tiền tố của mẫu (khởi đầu là rỗng).

Đầu ra: k tập mẫu phổ biến không chắc chắn có tiện ích cao đầu tiên

1. **Với** mỗi $P_x \in$ tree.header_table, **thực hiện**:
2. **Nếu** Px ∉ prefix, **thì**:
3. *// chiến lước 4*
4. *overestimate* ← max_prob(node(*Px))*prefix.expSup*
5. **Nếu** *overestimate* < threshold, **thì**:
6. Trả về topUHUFP, dừng thuật toán
7. **Kết thúc**
8. *new_prefix* ← Px ∪ prefix
9. *// chiến lược 5, 6*
10. Thu thập { new_prefix }-CPB và có được pu(*new_prefix*)
11. **Nếu** pu(*new_prefix*) >= minUtil, **thì**:
12. Tính expSup(new_prefix), Util(*new_prefix-tree*, minSup, minUtil, k, *new_prefix*)
13. // *chiến lược 7, 8*
14. Tạo cây UHUFP của *new_prefix* (*new_prefix -tree*)
15. Kiểm tra expSup và Util và topUHUFP ← topUHUFP ∪ *new_prefix*
16. Nâng ngưỡng khi đủ điều kiện
17. **Nếu** *new_prefix-tree* có các node con, **thì**:
18. TUHUFP_Growth(*new_prefix-tree*, minSup, minUtil, *new_prefix*)
19. **Kết thúc**
20. **Kết thúc**
21. **Kết thúc**

22. **Kết thúc**
23. Trả về topUHUFP

5.6 Hướng dẫn chạy chương trình

Chương trình thực nghiệm được lưu trữ trên hệ thống github với đường dẫn sau:

https://github.com/huynhphucnguyen/KLTN_Mining-top-k-high-utility-frequent-itemsets-from-uncertain-databases.git

Yêu cầu môi trường thực nghiệm:

Máy tính đã được cài đặt đầy đủ python và jupyter. Thực nghiệm trên Visual Studio Code.

Hướng dẫn thực hiện thực nghiệm:

1. Tải file nén về, tiến hành giải nén
2. Vào thư mục data giải nén file data.zip (extract here)
3. Mở folder đã được giải nén từ trước tại bước 1
4. Tải các môi trường/thư viện được đề xuất (nếu có).
5. Vào thư mục algorithm/src, mở file TUHUFPAlgo và TUHUFP_Growth:
   - TUHUFPAlgo: chương trình thuật toán TUHUFP
   - TUHUFPGrowth: chương trình thuật toán TUHUFP-Growth
6. Tại Cell chứa 2 thuật toán, nhấn Execute Cell (biểu tượng tam giác)
7. Kéo xuống các Cell dưới chương trình, có 3 Cell để thực nghiệm cho 3 bộ dữ liệu Foodmart, Chess và Retail. Dựa trên thay đổi về số liệu đầu vào, chương trình sẽ thực hiện theo số liệu được nhập:
   a. Thuật toán TUHUFP:

      Trong phương thức run_TUHUFP_algorithm có 2 trường thông tin cuối là ngưỡng tiện ích và số lượng top k mẫu cần lấy, điều chỉnh 2 trường thông tin này và nhấn Execute Cell để thực nghiệm thuật toán.

b. Thuật toán TUHUFP-Growth:

Điều chỉnh giá trị các trường k (top mẫu cần lấy) và percentage (ngưỡng tiện ích), sau đó nhấn Execute Cell để thực nghiệm thuật toán.

Danh sách các giá trị thực nghiệm cho các bộ dữ liệu:

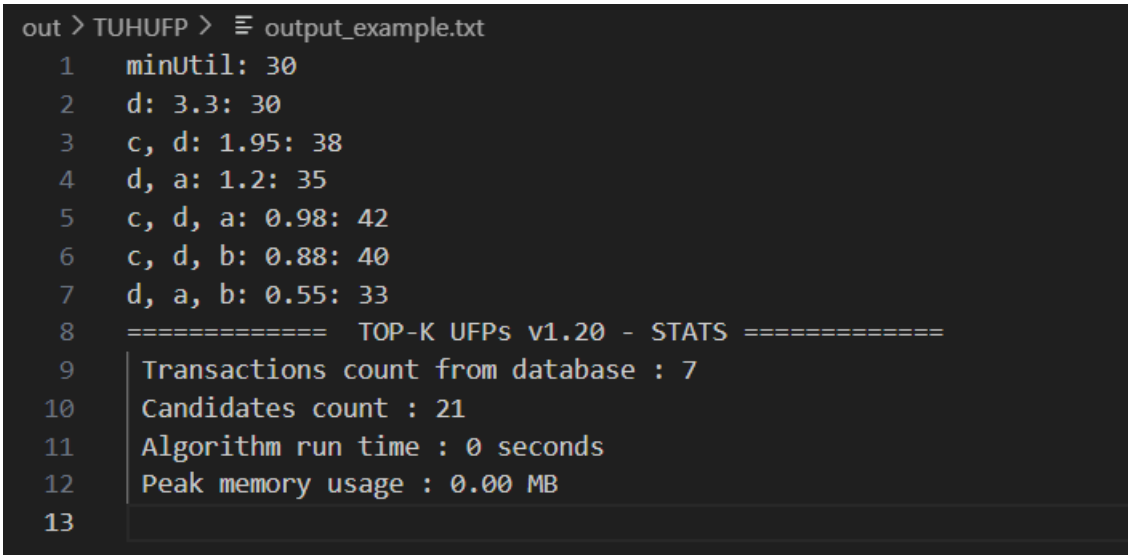
Bảng 5.1 Danh sách các giá trị thực nghiệm

Tên bộ dữ liệu	Ngưỡng tiện ích	Số lượng top k mẫu
Foodmart	0.0004	Top 100, 300, 500, 700, 900
Retail	0.0001	Top 100, 300, 500, 700, 900
Chess	0.001	Top 100, 300, 500, 700, 900

CHƯƠNG 6. THỰC NGHIỆM

6.1 Thực nghiệm với dữ liệu mẫu

Thực nghiệm đối với dữ liệu mẫu ở Bảng 3.1 đối với 2 thuật toán TUHUFP và TUHUFP-Growth, lấy ngưỡng tiện ích là 30% và số lượng top k là 6, kết quả ta nhận được đối với 2 thuật toán như sau:

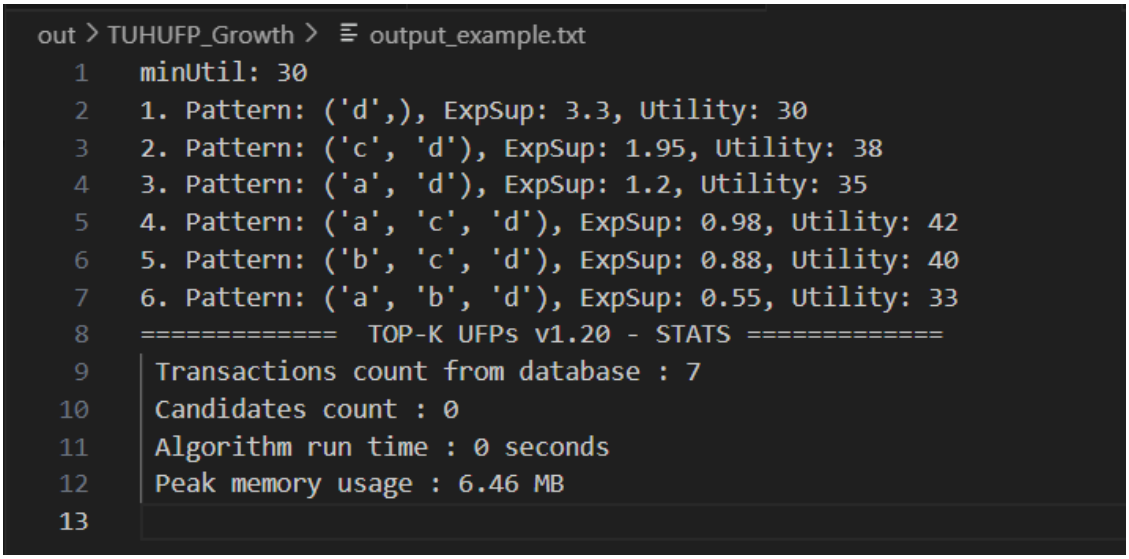


Hình 6.1 Kết quả chạy thực nghiệm bộ mẫu thuật toán TUHUFP

Từ CUP 1-mẫu như Hình 4.1 ta sẽ lấy được các UHUFP đầu tiên là <d, 3.3, 30> với giá trị tiện ích là 30 do đó d là một UHUFP, thêm d vào danh sách kết quả. Và danh sách mẫu tham gia gồm {c, d, a, b, f, e} với các TWU lần lượt là {84, 82, 57, 65, 45, 48} (đều lớn hơn min_util).

Tiếp theo, TUHUFP sẽ sử dụng mẫu c để ghép với các mẫu còn lại (d, a, b, f và e) trong danh sách mẫu tham gia. Mẫu <cd, 1.95, 38> được ghép đầu tiên, được thêm vào danh sách kết quả và danh sách mẫu tham gia với TWU bằng 69. Tiếp đến là <ca, 1.8, 22> vì ca có tiện ích nhỏ hơn min_util nên sẽ bị bỏ qua. Lần lượt cb, cf, ce đều có giá trị tiện ích nhỏ hơn min_util nên không thêm vào danh sách kết quả nhưng TWU lại có giá trị (60, 32, 35) lớn hơn min_util nên được thêm vào danh sách mẫu tham gia. Sau khi hết CUP 1-mẫu, ta đến vòng tiếp theo với các mẫu vừa được tạo trong danh sách mẫu tham gia (cd, ca, cb, cf, ce). Ở bước này, ta sẽ bắt đầu từ cd

và ca sau đó TUHUFP sẽ tiếp tục gọi **TUHUFP_Search** để tìm các mẫu còn lại, kết quả sau khi ghép mẫu tất cả các mẫu được thể hiện tại Hình 6.1. Threshold sẽ được cập nhật khi thỏa số lượng k trong danh sách kết quả .



Hình 6.2 Kết quả chạy thực nghiệm bộ mẫu thuật toán TUHUFP-Growth

Từ cây UHUFP tại Hình 4.3, ta bắt đầu với {c} tuy nhiên {c} là một mẫu không hứa hẹn do giá trị tiện ích của c trong path nhỏ hơn minUtil (18 < 30), nên ta bỏ qua {c}. Với {d} dựa vào cây UHUFP ta thu thập được 5 path cho mẫu {d}, gọi là {d}-CPB gồm: ('c', 0.9, 1, 6), ('d', 0.6, 5, 6); ('c', 0.7, 3, 8), ('d', 0.6, 5, 8); ('c', 0.8, 6, 11), ('d', 0.9, 5, 11); ('c', 0.9, 3, 13), ('d', 0.3, 10, 13); ('d', 0.9, 5, 5). Ta thấy được {d} là một mẫu hứa hẹn với giá trị tiện ích cục bộ là 43 (> minUtil) có 4 path bao gồm: ('c', 0.9, 1, 6), ('d', 0.6, 5, 6); ('c', 0.7, 3, 8), ('d', 0.6, 5, 8); ('c', 0.8, 6, 11), ('d', 0.9, 5, 11); ('c', 0.9, 3, 13), ('d', 0.3, 10, 13); tính được giá trị độ hỗ trợ mong đợi của {d} là 3.3 và giá trị tiện ích là 30. Thêm {d} và danh sách kết quả. Xây dựng {d}-tree (UHUFP cục bộ của {d}) từ {d}-CPB. Tiếp tục gọi TUHUFP_Growth với {d}-tree, duyệt header table của {d}-tree, ta có được mẫu tiếp theo là {c, d}, thu thập {c, d}-CPB và biết được {c, d} là mẫu hứa hẹn với pu({c,d}) = 38, ta có được mẫu {c, d} với expSup = 1.95, utility = 38 (thêm vào danh sách kết quả) và tạo {c, d}-tree cho bước tiếp theo. Tuy nhiên {c, d}-tree rỗng, ta kết thúc khai thác cây cục bộ của

{d} tại đây. Ta tiếp tục với bảng header table toàn cục với các mẫu: {a}, {b}, {f}. Tiếp tục thuật toán cho tới khi các mẫu được duyệt hết và threshold sẽ được cập nhật khi số lượng k được thỏa mãn, ta có được kết quả như Hình 6.2.

Từ Hình 6.1 và 6.2, ta có thể thấy hai thuật toán cho ra kết quả chính xác và giống nhau, đảm bảo tính đúng đắn đối với chương trình.

6.2 Thiết lập thực nghiệm

Thực nghiệm được thực hiện trên Laptop Acer với cầu hình Intel Core i7-11800H, bộ nhớ RAM 16GB và hệ điều hành Windows 11. Thuật toán được cài đặt bằng ngôn ngữ Python trên Visual Studio Code.

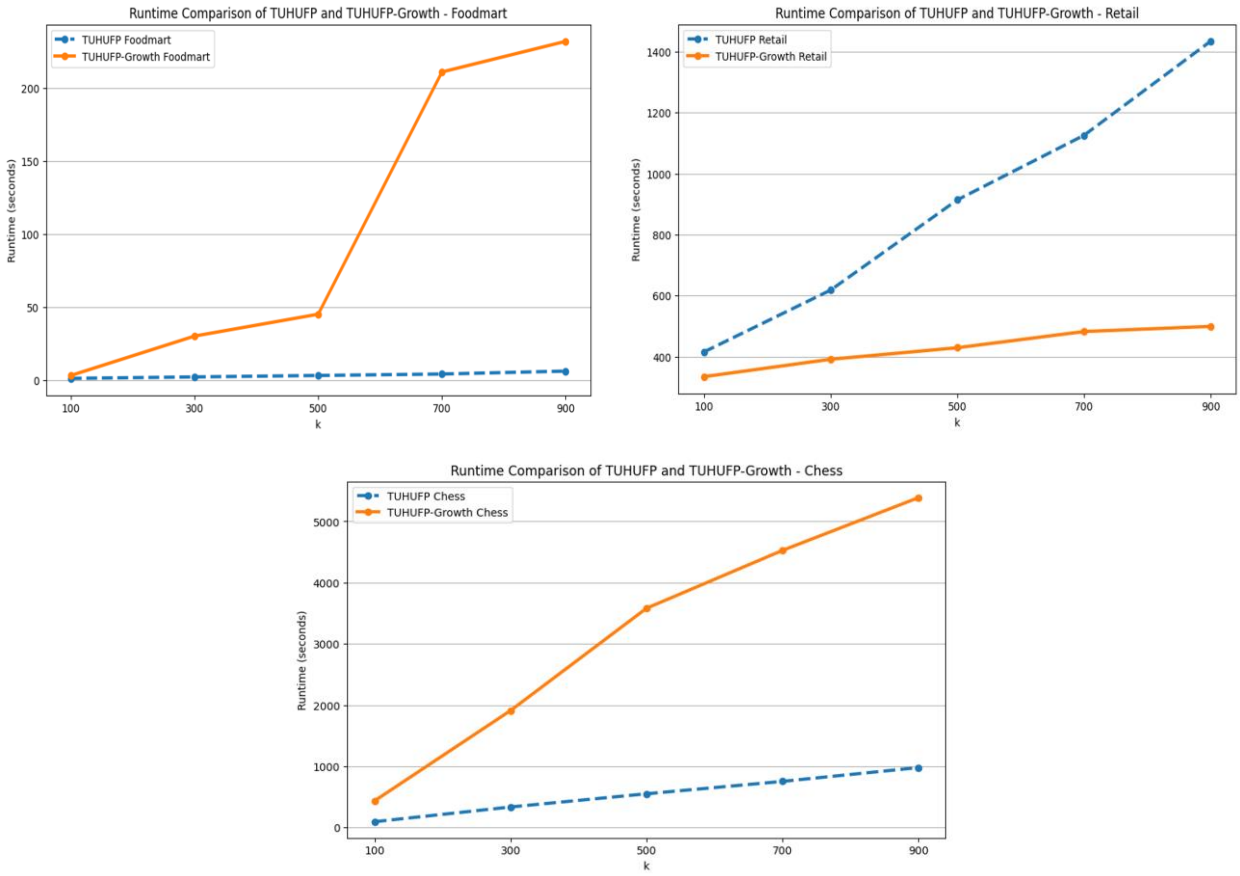
Các cơ sở dữ liệu được sử dụng trong thực nghiệm này gồm: Foodmart, Retail, Chess. Đây là các cơ sở dữ liệu phổ biến được sử dụng nhiều cho việc thực nghiệm khai thác dữ liệu và được tham khảo trên thư viện nguồn dữ liệu mở SPMF: https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php#d3

6.3 Kết quả thực nghiệm

Dựa trên số liệu đã cung cấp ở Bảng 5.1, ta có được các kết quả thực nghiệm sau:

6.3.1 Thời gian khai thác

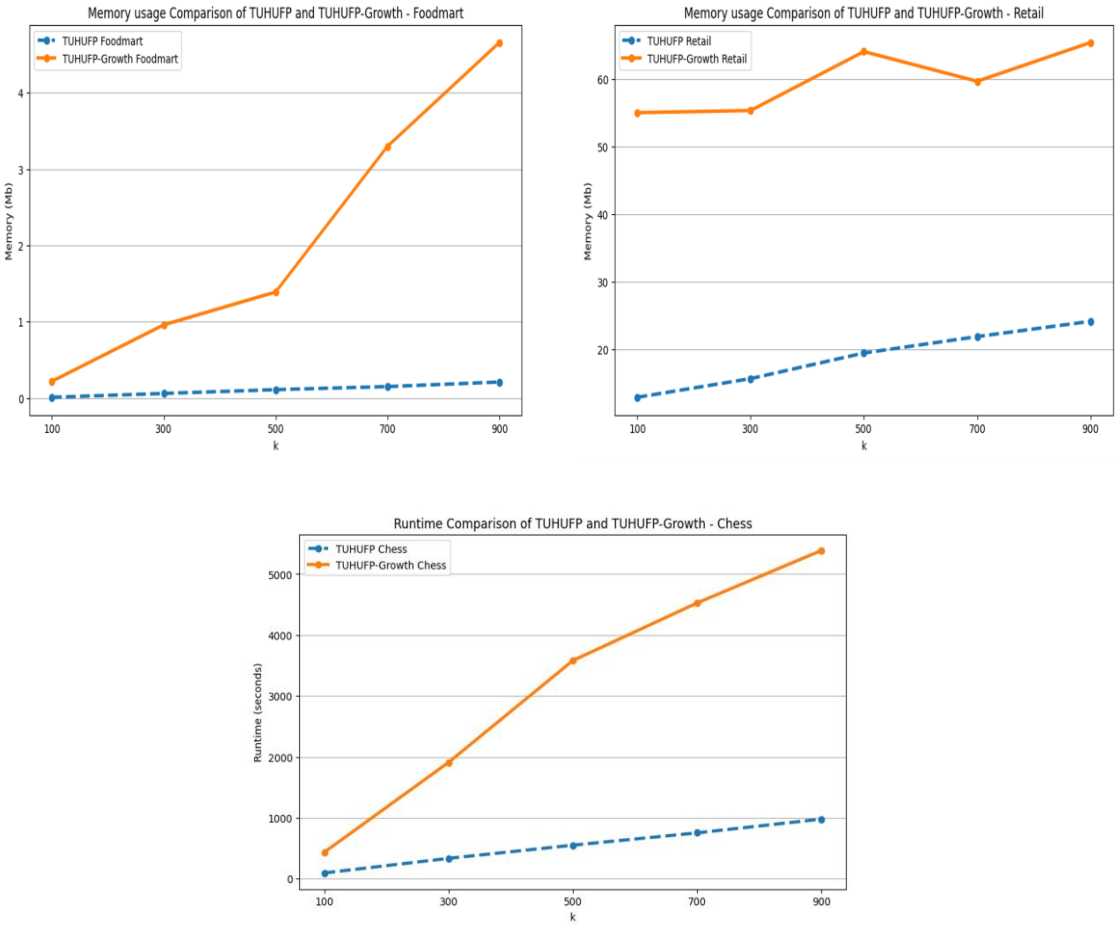
Hình 6.1 thể hiện thời gian khai thác của thuật toán TUHUFP và TUHUFP-Growth trên ba bộ cơ sở dữ liệu với các mức k khác nhau. Kết quả cho thấy ở hai bộ dữ liệu Foodmart và Chess, thời gian thực thi của thuật toán TUHUFP tốt hơn khá nhiều so với thuật toán TUHUFP-Growth. Tuy nhiên ở bộ dữ liệu Retail, thuật toán TUHUFP-Growth đã cho ra thời gian xử lý tốt hơn. Xét trên quy mô của ba bộ dữ liệu, Retail có quy mô lớn nhất. Từ đó có thể suy ra thuật toán TUHUFP xử lý tốt đối với các bộ dữ liệu quy mô vừa và nhỏ, trong khi TUHUFP-Growth sẽ xử lý tốt đối với các bộ dữ liệu quy mô lớn.



Hình 6.1 Thời gian khai thác của TUHUFP và TUHUFP-Growth trên các điều kiện khác nhau

6.3.2 Bộ nhớ sử dụng

Hình 6.2 thể hiện bộ nhớ sử dụng của thuật toán TUHUFP và TUHUFP-Growth trên ba bộ dữ liệu Foodmart, Chess, Retail. Có thể thấy thuật toán TUHUFP có sự tối ưu trong sử dụng bộ nhớ tốt hơn nhiều so với TUHUFP-Growth ở cả ba bộ dữ liệu. Từ đó có thể kết luận rằng cấu trúc dữ liệu CUP-list có phần nhỉnh hơn so với cấu trúc UHUFP-tree.



Hình 6.2 Bộ nhớ sử dụng của TUHUFP và TUHUFP-Growth trên các điều kiện khác nhau

CHƯƠNG 7. KẾT LUẬN

Thuật toán TUHUFP và TUHUFP-Growth là những công cụ mạnh mẽ trong việc khai thác các tập mẫu phổ biến tiện ích cao từ các cơ sở dữ liệu không chắc chắn. TUHUFP đã khẳng định tính hiệu quả của nó trong việc xử lý các cơ sở dữ liệu phức tạp với khả năng khám phá k tập mẫu phổ biến có độ tiện ích cao một cách nhanh chóng và chính xác. Thuật toán này giúp thu hẹp không gian tìm kiếm, giảm bớt các mẫu không cần thiết và đảm bảo rằng chỉ những tập mẫu quan trọng nhất mới được khai thác.

TUHUFP-Growth, với cơ chế cải tiến từ việc sử dụng cấu trúc cây UHUFP, cung cấp một phương pháp tiếp cận tối ưu hơn. Nó không chỉ giảm đáng kể không gian lưu trữ mà còn cải thiện tốc độ xử lý, đặc biệt trong các cơ sở dữ liệu có quy mô lớn và chứa nhiều thông tin không chắc chắn. TUHUFP-Growth tận dụng được các chiến lược như loại bỏ các mục không hứa hẹn và giảm thiểu tiện ích của các nút con, từ đó tối ưu hóa quá trình khai thác và tăng cường hiệu quả của toàn bộ quá trình.

Sự kết hợp giữa hai thuật toán TUHUFP và TUHUFP-Growth mang lại nhiều ứng dụng thực tiễn, giúp phát hiện các mẫu tiềm năng, đưa ra các dự đoán chính xác hơn và hỗ trợ các quyết định chiến lược dựa trên dữ liệu. Trong tương lai, việc cải tiến thêm các chiến lược xử lý dữ liệu không chắc chắn sẽ giúp các thuật toán này ngày càng hoàn thiện hơn, mở ra nhiều cơ hội cho nghiên cứu và ứng dụng thực tiễn trong nhiều lĩnh vực.

TÀI LIỆU THAM KHẢO

[1] C. Aggarwal, Y. L. (2009). *Frequent pattern mining with uncertain data.* Knowledge Discovery and Data Mining.

[2] C. Leung, M. M. (2008). *A Tree-Based Approach for Frequent Pattern Mining from Uncertain Data.* Pacific-Asia Conference on Knowledge Discovery and Data Mining.

[3] Dam, T.-L., Li, K., Fournier-Viger, P., & Duong, Q.-h. (2016). *An efficient algorithm for mining top-rank-k frequent patterns.* Applied intelligence (Boston).

[4] Davashi, R. (2021). *ILUNA: Single-pass incremental method for uncertain frequent pattern mining without false positives.* Information Sciences.

[5] Davashi, R. (2022). *ITUFP: A fast method for interactive mining of Top-K frequent patterns from uncertain data.* Expert systems with applications.

[6] Deng, Z. (2014). *Fast mining Top-Rank-k frequent patterns by using Node-list.* Expert systems with applications.

[7] DŨNG, N. Đ. (2019). *KHAI THÁC TẬP MỤC LỢI ÍCH CAO SỬ DỤNG PHƯƠNG PHÁP TỐI ƯU ĐÀN KIẾN.* TRƯỜNG ĐẠI HỌC CÔNG NGHỆ.

[8] Duong, Q.-h., Liao, B., Fournier-Viger, P., & Dam, T.-L. (2016). *An efficient algorithm for mining the top-k high utility itemsets, using novel threshold raising and pruning strategies.* Knowledge-Based Systems.

[9] Fournier-Viger, P., Lin, J. C.-W., Chi, T. T., & Nkambou, R. (2019). *A Survey of High Utility Itemset Mining.* Springer.

[10] Grahne, G., & Zhu, J. (2005). *Fast algorithms for frequent itemset mining using FP-trees.* IEEE Transactions on Knowledge and Data Engineering.

[11] HAN, J., PEI, J., YIN, Y., & MAO, R. (2004). *Mining frequent patterns without candidate* . Data Mining and Knowledge Discovery.

[12] Huynh-Thi-Le, Q., Le, T., Vo, B., & Le, H. (2015). *An efficient and effective algorithm for mining top-rank-k frequent patterns.* Expert systems with applications.

[13] Krishnamoorthy, S. (2015). *Pruning strategies for mining high utility itemsets.* Expert systems with applications.

[14] Lan, G.-C., Hong, T.-P., & Tseng, V. (2012). *An efficient projection-based indexing approach for mining high utility itemsets.* Knowledge and Information Systems.

[15] Le, T., Vo, B., & Baik, S. (2018). *Efficient algorithms for mining top-rank-k erasable patterns using pruning strategies and the subsume concept.* Engineering applications of artificial intelligence.

[16] Le, T., Vo, B., Huynh, V.-N., Nguyen, N., & Baik, S. (2020). *Mining top-k frequent patterns from uncertain databases.* Springer.

[17] Lee, G., & Yun, U. (2017). *A new efficient approach for mining uncertain frequent patterns using minimum data structure without false positives.* Future generations computer systems.

[18] Lee, G., Yun, U., & Ryang, H. (2015). *An uncertainty-based approach: Frequent itemset mining from uncertain data with different item importance.* Knowledge-Based Systems.

[19] Lin, C.-W., Gan, W., Fournier-Viger, P., Hong, T.-P., & Tseng, V. (2016). *Efficient algorithms for mining high-utility itemsets in uncertain databases.* Knowledge-Based Systems.

[20] Lin, C.-W., Gan, W., Fournier-Viger, P., Hong, T.-P., & Tseng, V. (2016). *Efficient Mining of Uncertain Data for High-Utility Itemsets.* Interational Conference on Web-Age Information Management.

[21] Lin, C.-W., Gan, W., Fournier-Viger, P., Hong, T.-P., & Zhan, J. (2016). *Efficient mining of high-utility itemsets using multiple minimum utility thresholds.* Knowledge-Based Systems.

[22] Lin, C.-W., Hong, T.-P., & Lu, W.-H. (2011). *An effective tree structure for mining high utility itemsets.* Expert systems with applications.

[23] Lin, J. C.-W., Li, T., Fournier-Viger, P., & Su, T.-P. H.-H. (2016). *Efficient Mining of High Average-Utility Itemsets with Multiple Minimum Thresholds.* SpringerLink.

[24] Liu, Y., Liao, W., & Choudhary, A. (2005). *A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets.* Pacific-Asia Conference on Knowledge Discovery and Data Mining.

[25] Liu, Y.-H. (2015). *Mining time-interval univariate uncertain sequential pattern.* Data & Knowledge Engineering.

[26] Palacios, A. M., Martínez, A., Sánchez, L., & Couso, I. (2015). *Sequential pattern mining applied to aeroengine condition monitoring with uncertain health data.* Engineering applications of artificial intelligence.

[27] Qu, J.-F., & Fournier-Viger, M. L. (2019). *Efficient Algorithms for High Utility Itemset Mining Without Candidate Generation.* Springer.

[28] R. Agrawal, R. S. (1998). *Fast algorithms for mining association.*

[29] Ryang, H., & Yun, U. (2015). *Top-k high utility pattern mining with effective threshold raising strategies.* Knowledge-Based Systems.

[30] Tseng, V., Wu, C.-W., Fournier-Viger, P., & Yu, P. S. (2016). *Efficient Algorithms for Mining Top-K High Utility Itemsets.* IEEE Transactions on Knowledge and Data Engineering.

[31] V. Tseng, C.-W. W.-E. (2010). *UP-Growth: An Efficient Algorithm for High Utility Itemset Mining.* Knowledge Discovery and Data Mining.

[32] VU, V. V., LAM, M. T., T. T., MANH, L. T., NGUYEN, T. T., NGUYEN, L. V., . . . (Senior Member, I. A. (2023). *FTKHUIM: A Fast and Efficient Method for Mining Top-K High-Utility Itemsets.* IEEE Transactions on Knowledge and Data Engineering.

[33] Wu, C. W., Shie, B.-E., Yu, P. S., & Tseng, V. S. (2024). *Mining Top-K High Utility Itemsets.*

[34] Zaki, M. J., & Hsiao, C.-J. (2005). *Efficient algorithms for mining closed itemsets and their lattice structure.* IEEE Transactions on Knowledge and Data Engineering.