Thiết kế và phân tích các thuật toán Tìm kiếm các tập mục hữu ích cao Top-K từ cơ sở dữ liệu không chắc chắn

(Design and analysis of algorithms interactively searching for Top-K high-utility itemsets from uncertain databases)

TS. Nguyễn Chí Thiện*, Hà Trọng Nguyễn†, Lương Chí Trung‡, Đỗ Thị Kiều Thanh§
*Giảng viên hướng dẫn, Trường Đại học Tôn Đức Thắng, Việt Nam
†MSSV: 52200148, ‡MSSV: 52200166, §MSSV: 52200144
Khoa Công nghệ Thông Tin, Trường Đại học Tôn Đức Thắng, Việt Nam

***Tóm tắt nội dung*—Báo cáo này trình bày việc cải tiến thuật toán ITUFP [4] (*Interactive Top-K Uncertain Frequent Pattern Mining*), một phương pháp ban đầu được thiết kế để khai thác các mẫu thường xuyên Top-K (*frequent patterns*) từ cơ sở dữ liệu không chắc chắn (Uncertain Databases - UDB) [2], thành thuật toán mới nhằm khai thác các tập mục hữu ích cao nhất (Top-K High-Utility Itemsets - HUIs) [3]. Trong UDB, mỗi mục không chỉ có xác suất xuất hiện mà còn được gắn với giá trị hữu ích, dẫn đến việc mở rộng từ bài toán tìm kiếm mẫu thường xuyên sang bài toán khai thác HUIs, vốn phức tạp hơn nhiều.**

Thuật toán được cải tiến tập trung vào việc xây dựng cấu trúc dữ liệu hiệu quả gồm UPList và IMCUPList, cho phép tổ hợp thông tin các mục, giảm số lần quét cơ sở dữ liệu và áp dụng chiến lược ngắt ngưỡng (early pruning) để loại bỏ các tập mục không tiềm năng. Ngoài ra, quy trình khai thác Top-K HUIs được tối ưu hóa để hỗ trợ tốt trong môi trường tương tác với các tham số thay đổi.

Kết quả thử nghiệm trên các tập dữ liệu thực tế như FoodMart, Retail, ... cho thấy thuật toán đạt hiệu suất vượt trội về thời gian chạy và bộ nhớ sử dụng so với các phương pháp trước đây như LUNA [5] và ITUFP gốc. Bằng cách cải tiến từ ITUFP, thuật toán đề xuất không chỉ mở rộng phạm vi ứng dụng từ mẫu thường xuyên sang HUIs, mà còn chứng minh tiềm năng ứng dụng lớn trong phân tích dữ liệu, gợi ý sản phẩm, và tối ưu hóa chuỗi cung ứng.

I. Giới thiệu

Trong lĩnh vực khai phá dữ liệu, bài toán tìm kiếm các tập mục hữu ích cao nhất (*High-Utility Itemsets* - HUIs) đã nhận được sự quan tâm đáng kể do tiềm năng ứng dụng lớn trong các lĩnh vực như phân tích giỏ hàng, hệ thống gợi ý sản phẩm, và quản lý chuỗi cung ứng. Khác với các bài toán khai thác mẫu thường xuyên (*frequent patterns*), HUIs không chỉ xem xét tần suất xuất hiện của các mục mà còn tính đến giá trị hữu ích (*utility*) của chúng trong cơ sở dữ liệu. Điều này làm cho bài toán khai thác HUIs trở nên phức tạp hơn, đặc biệt khi xử lý các cơ sở dữ liệu không chắc chắn (*Uncertain Databases* - UDB).

A. Mục tiêu nghiên cứu

Mục tiêu của nghiên cứu này là phát triển một thuật toán hiệu quả để khai thác các tập mục hữu ích cao nhất từ UDB, trong đó mỗi mục không chỉ có xác suất xuất hiện mà còn gắn với giá trị hữu ích. Thuật toán được đề xuất, dựa trên cấu trúc và nguyên lý của ITUFP (*Interactive Top-K Uncertain Frequent Pattern Mining*), đã được cải tiến để giải quyết bài toán HUIs thay vì các mẫu thường xuyên.

B. Ý nghĩa của bài toán

Khai thác HUIs từ UDB mang lại nhiều lợi ích thực tế:

- **Phân tích giỏ hàng:** Tìm các tổ hợp sản phẩm có giá trị cao trong các giao dịch bán lẻ, hỗ trợ tối ưu hóa chiến lược kinh doanh.
- **Hệ thống gợi ý:** Đưa ra các gợi ý phù hợp dựa trên các sản phẩm có giá trị cao mà khách hàng thường xuyên mua.
- **Quản lý chuỗi cung ứng:** Tối ưu hóa việc phân phối và lưu kho dựa trên các mục có giá trị cao và xuất hiện thường xuyên trong giao dịch.

C. Thách thức trong khai thác HUIs từ UDB

Khai thác các tập mục hữu ích cao nhất (HUIs) từ cơ sở dữ liệu không chắc chắn (UDB) đặt ra nhiều thách thức lớn, bao gồm:

- **Độ phức tạp tính toán:** Việc khai thác HUIs yêu cầu tính toán giá trị hữu ích tổng hợp của các tổ hợp mục (*itemset*) trong từng giao dịch, sau đó tổng hợp trên toàn bộ cơ sở dữ liệu. Với số lượng giao dịch và mục lớn, quá trình này có thể tiêu tốn rất nhiều tài nguyên tính toán.
- **Bùng nổ tổ hợp [12]:** Khi số lượng mục trong cơ sở dữ liệu tăng, số lượng tổ hợp mục cần xem xét sẽ tăng theo cấp số nhân. Chẳng hạn, với n mục, số tổ hợp khả dĩ là $2^n - 1$, gây ra tình trạng "bùng nổ tổ hợp". Đặc biệt, trong UDB, mỗi tổ hợp mục còn phải tính thêm xác suất tồn tại và giá trị hữu ích, làm tăng đáng kể khối lượng tính toán.
- **Tối ưu hóa bộ nhớ:** Để lưu trữ thông tin cần thiết cho việc tính toán giá trị hữu ích và xác suất kỳ vọng, các thuật toán thường sử dụng các cấu trúc dữ liệu như danh sách UPLists và IMCUPLists. Tuy nhiên, với cơ sở dữ liệu lớn, việc quản lý bộ nhớ hiệu quả trở thành một thách thức lớn.
- **Ngắt ngưỡng hiệu quả: [13]** Một thách thức quan trọng khác là áp dụng các chiến lược ngắt ngưỡng (early pruning) để loại bỏ các tổ hợp mục không tiềm năng, từ đó giảm số lượng tổ hợp phải xem xét mà không làm mất các tập mục hữu ích cao.

1) Ví dụ về bùng nổ tổ hợp: Xét một cơ sở dữ liệu đơn giản chứa 5 mục (x_1, x_2, x_3, x_4, x_5). Số lượng tổ hợp khả dĩ có thể được tạo ra từ các mục này là:

$$2^5 - 1 = 31.$$

Nếu số lượng mục tăng lên 10, số lượng tổ hợp sẽ là:

$$2^{10} - 1 = 1023.$$

Trong các cơ sở dữ liệu thực tế với hàng nghìn mục, số lượng tổ hợp khả dĩ sẽ tăng lên đến hàng triệu hoặc hàng tỷ. Khi mỗi tổ hợp cần tính toán giá trị hữu ích và xác suất kỳ vọng trên toàn bộ các giao dịch trong UDB, chi phí tính toán sẽ trở nên không khả thi nếu không áp dụng các phương pháp tối ưu.

2) Giải pháp giảm thiểu bùng nổ tổ hợp: Thuật toán được đề xuất trong nghiên cứu này sử dụng:

- **UPList và IMCUPList:** Hai cấu trúc dữ liệu này giúp lưu trữ thông tin tổ hợp một cách hiệu quả, cho phép tính toán giá trị hữu ích và xác suất kỳ vọng mà không cần truy cập trực tiếp vào toàn bộ cơ sở dữ liệu.
- **Chiến lược ngắt ngưỡng (early pruning):** Loại bỏ các tổ hợp mục không tiềm năng dựa trên giá trị ngưỡng tối thiểu (*minimum threshold*), từ đó giảm đáng kể số lượng tổ hợp cần xem xét.

D. Đóng góp chính của thuật toán

Thuật toán này đóng góp các cải tiến quan trọng trong việc khai thác HUIs từ UDB:

- **Cải tiến thuật toán ITUFP:** Thuật toán ban đầu được thiết kế để tìm kiếm các mẫu thường xuyên đã được mở rộng và cải tiến để tìm kiếm các tập mục hữu ích cao.
- **Cấu trúc dữ liệu tối ưu:** Sử dụng UPList để lưu trữ thông tin mục và IMCUPList để tổ hợp dữ liệu, giảm đáng kể số lần quét cơ sở dữ liệu.
- **Chiến lược ngắt ngưỡng (early pruning):** Áp dụng ngắt ngưỡng để loại bỏ các tổ hợp mục không tiềm năng, giúp tối ưu hóa thời gian xử lý.
- **Hiệu quả thực nghiệm:** Phân tích chi tiết hiệu suất của thuật toán trên các tập dữ liệu thực tế như FoodMart, Retail so sánh với các thuật toán trước đây như LUNA và ITUFP gốc.

E. Cấu trúc bài báo

Phần còn lại của báo cáo được tổ chức như sau:

- **Phần II: Các công trình liên quan** - Trình bày tổng quan các nghiên cứu trước đây về khai thác HUIs và các phương pháp liên quan (II).
- **Phần III: Định nghĩa bài toán** - Định nghĩa các khái niệm cơ bản như giá trị hữu ích, xác suất, và các chỉ số kỳ vọng trong UDB (III).
- **Phần IV: Thuật toán ITUFP cải tiến** - Giới thiệu chi tiết cấu trúc dữ liệu, mã giả, và quy trình thuật toán (IV).
- **Phần V: Đánh giá độ phức tạp thời gian** - Phân tích lý thuyết về độ phức tạp thời gian của thuật toán ITUFP (V).
- **Phần VI: Phân tích kết quả thực nghiệm** - Phân tích kết quả thử nghiệm trên các tập dữ liệu thực tế (VI).
- **Phần VII: Kết luận và hướng phát triển** - Tóm tắt kết quả nghiên cứu và đề xuất các hướng mở rộng (VII).

II. Các Công Trình Liên Quan

A. Tổng quan về khai thác HUIs và dữ liệu không chắc chắn

Khai thác tập mục hữu ích cao (HUIs) từ dữ liệu không chắc chắn (Uncertain Databases - UDB) là một trong những hướng nghiên cứu quan trọng trong lĩnh vực khai phá dữ liệu. UDB chứa các giao dịch trong đó mỗi mục có giá trị hữu ích và xác suất tồn tại, gây ra nhiều thách thức trong việc xử lý và khai thác thông tin. Các thuật toán khai thác HUIs tập trung vào việc tìm kiếm các tập mục có giá trị hữu ích cao nhất, vượt qua một ngưỡng nhất định, dựa trên cả giá trị và xác suất tồn tại.

B. Các phương pháp nghiên cứu trước đây

1) Thuật toán LUNA: LUNA (List-based Uncertain frequent pattern mining) là một thuật toán dựa trên cấu trúc CUP- List, thuật toán này tận dụng tính chất cắt tỉa (pruning) để giảm thiểu không gian tìm kiếm, từ đó cải thiện hiệu suất xử lý trên các cơ sở dữ liệu không chắc chắn. Tuy nhiên, LUNA vẫn gặp hạn chế khi xử lý các cơ sở dữ liệu lớn và phức tạp, đặc biệt khi số lượng tổ hợp mục tăng nhanh.

2) Thuật toán TUFP: TUFP (Top-K Utility Frequent Patterns) là một thuật toán khai thác các mẫu thường gặp Top-K từ cơ sở dữ liệu không chắc chắn. Thuật toán này áp dụng các chiến lược nâng ngưỡng (*early threshold raising*) để giảm số lượng mẫu ứng viên và tăng hiệu suất khai thác. TUFP đã chứng minh tính hiệu quả trong việc giảm thời gian chạy và sử dụng bộ nhớ, nhưng chưa tối ưu hóa hoàn toàn khi xử lý với dữ liệu có giá trị hữu ích cao.

3) Thuật toán UHUOPM: [7] UHUOPM (Uncertain High-Utility Occupancy Pattern Mining) khai thác các mẫu chiếm dụng hữu ích cao trong cơ sở dữ liệu không chắc chắn, sử dụng các cấu trúc như PUO-List và PFU-Table. Thuật toán này kết hợp ba yếu tố: tần suất, xác suất, và giá trị hữu ích, giúp cải thiện đáng kể hiệu suất khai thác. Tuy nhiên, UHUOPM yêu cầu tài nguyên bộ nhớ lớn khi xử lý dữ liệu phức tạp và không mở rộng tốt khi số lượng giao dịch tăng.

4) Các phương pháp truyền thống: Các thuật toán truyền thống như Apriori và FP-Growth đã được sử dụng rộng rãi để khai thác các tập mục thường gặp và tập mục hữu ích cao từ dữ liệu xác định. Tuy nhiên, các phương pháp này không thể xử lý tính không chắc chắn trong UDB, do không tính đến xác suất tồn tại của các mục trong giao dịch.

C. Hạn chế trong các nghiên cứu trước đây

Các nghiên cứu trước đây về khai thác HUIs từ UDB còn tồn tại một số hạn chế:

- **Khả năng mở rộng kém:** Nhiều thuật toán không tối ưu khi xử lý cơ sở dữ liệu lớn hoặc dữ liệu có số lượng tổ hợp mục cao.
- **Tối ưu hóa tài nguyên:** Một số thuật toán tiêu tốn quá nhiều bộ nhớ hoặc thời gian xử lý.
- **Thiếu tính tổng hợp:** Chưa kết hợp hiệu quả các yếu tố như xác suất, giá trị hữu ích và ngưỡng cắt tỉa.
- **Thiếu hỗ trợ Top-K:** Nhiều thuật toán chỉ tập trung vào ngưỡng tối thiểu (min-utility), chưa tối ưu hóa khai thác Top-K HUIs.

D. Đóng góp chính của bài báo

Để giải quyết các hạn chế trên, bài báo này đề xuất:

- **Thuật toán ITUFP cải tiến:** Thuật toán khai thác Top-K HUIs từ UDB, được phát triển dựa trên ITUFP.
- **Sử dụng IMCUP-List:** Một cấu trúc dữ liệu hiệu quả, giúp giảm thiểu bùng nổ tổ hợp và tăng tốc độ xử lý.
- **Chiến lược ngắt ngưỡng hiệu quả:** Áp dụng các chiến lược ngắt ngưỡng thông minh để loại bỏ các tổ hợp mục không tiềm năng.
- **Phân tích so sánh:** Đánh giá hiệu suất thuật toán qua các thử nghiệm thực tế, so sánh với các phương pháp trước đây như LUNA và TUFP.

III. Định nghĩa bài toán

A. Cơ sở dữ liệu không chắc chắn (UDB)

Cơ sở dữ liệu không chắc chắn (Uncertain Database - UDB) là tập hợp các giao dịch, trong đó mỗi giao dịch bao gồm:

- **Items:** Các mục x_i xuất hiện trong giao dịch.
- **Utilities:** Giá trị hữu ích $u(x_i)$ của từng mục.
- **Probabilities:** Xác suất $P(x_i, T_j)$ tồn tại của mục trong giao dịch T_j.

Dữ liệu mẫu:

```
items, item_utilities, item_probabilities
1 3, 10 30, 0.7 0.1
1 2 3 4, 40 20 30 50, 0.4 0.4 0.5 0.9
2 3 4, 20 30 10, 0.6 0.3 0.2
3 4, 30 40, 0.4 0.4
1 2, 50 30, 1 0.7
3 4, 20 60, 0.2 0.8
1 2 3, 30 40 50, 0.9 0.9 0.9
```

Mỗi giao dịch bao gồm danh sách các mục, giá trị hữu ích và xác suất tương ứng.

B. UPList: Cấu trúc dữ liệu đầu tiên

UPList là cấu trúc lưu trữ thông tin của từng mục x_i từ cơ sở dữ liệu UDB, được xây dựng từ các giao dịch. Mỗi UPList chứa:

- **TID:** ID của giao dịch chứa x_i.
- **Probability:** $P(x_i, T_j)$, xác suất tồn tại của x_i trong giao dịch T_j.
- **Utility:** $u(x_i)$, giá trị hữu ích của x_i.

Từ UPList, hai chỉ số quan trọng được tính:

- **Total Utility (TU):** Tổng giá trị hữu ích của mục x_i trên tất cả các giao dịch:

$$\text{Total Utility} = \sum_{T_j} u(x_i, T_j).$$

- **Expected Support (expSup):** Tổng xác suất tồn tại của x_i trên tất cả các giao dịch:

$$\text{Expected Support} = \sum_{T_j} P(x_i, T_j).$$

1) Ví dụ: UPList của $x = 1$: Dữ liệu mẫu cho mục 1:

```
TID, Utility, Probability
1, 10, 0.7
2, 40, 0.4
5, 50, 1.0
7, 30, 0.9
```

Từ đây:

$$UPList(1) = [(1, 0.7, 10), (2, 0.4, 40), (5, 1.0, 50), (7, 0.9, 30)]$$

Tính toán các chỉ số:

- **Total Utility:**

$$\text{Total Utility} = 10 + 40 + 50 + 30 = 130$$

- **Expected Support:**

$$\text{ExpSup} = 0.7 + 0.4 + 1.0 + 0.9 = 3.0$$

2) UPList của các mục khác:

- $UPList(2)$:

$$UPList(2) = [(2, 0.4, 20), (3, 0.6, 20), (5, 0.7, 30), (7, 0.9, 40)]$$

Total Utility: $20 + 20 + 30 + 40 = 110$
ExpSup: $0.4 + 0.6 + 0.7 + 0.9 = 2.6$

- $UPList(3)$:

$$UPList(3) = [(1, 0.1, 30), (2, 0.5, 30), (3, 0.3, 30), (4, 0.4, 30), (6, 0.2, 20), (7, 0.9, 50)]$$

Total Utility: $30 + 30 + 30 + 30 + 20 + 50 = 190$
ExpSup: $0.1 + 0.5 + 0.3 + 0.4 + 0.2 + 0.9 = 2.4$

- $UPList(4)$:

$$UPList(4) = [(2, 0.9, 50), (3, 0.2, 10), (4, 0.4, 40), (6, 0.8, 60)]$$

Total Utility: $50 + 10 + 40 + 60 = 160$
ExpSup: $0.9 + 0.2 + 0.4 + 0.8 = 2.3$

C. IMCUPList: Kết hợp từ UPLists hoặc IMCUPLists

IMCUPList là cấu trúc dữ liệu mở rộng từ UPLists hoặc IMCUPLists để lưu trữ thông tin các tổ hợp mục. Mỗi IMCUPList chứa:

- **TID:** ID của giao dịch chứa tổ hợp mục.
- **Combined Probability:** Xác suất kết hợp của tổ hợp mục:

$$P(X, T_j) = \prod_{x \in X} P(x, T_j).$$

- **Combined Utility:** Giá trị hữu ích kết hợp của tổ hợp mục:

$$u(X, T_j) = \sum_{x \in X} u(x, T_j).$$

1) Ví dụ: IMCUPList của $\{1, 2\}$: Kết hợp $UPList(1)$ và $UPList(2)$:

$$UPList(1) = [(1, 0.7, 10), (2, 0.4, 40), (5, 1.0, 50), (7, 0.9, 30)].$$

$$UPList(2) = [(2, 0.4, 20), (3, 0.6, 20), (5, 0.7, 30), (7, 0.9, 40)].$$

a) Quy trình tính toán:: Các giao dịch có trong cả hai danh sách T_2, T_5, T_7:

- T_2: $P(1,T_2)\cdot P(2,T_2)=0.4\cdot 0.4=0.16$, $u(1,T_2)+u(2,T_2)=40+20=60$
- T_5: $P(1,T_5)\cdot P(2,T_5)=1.0\cdot 0.7=0.7$, $u(1,T_5)+u(2,T_5)=50+30=80$
- T_7: $P(1,T_7)\cdot P(2,T_7)=0.9\cdot 0.9=0.81$, $u(1,T_7)+u(2,T_7)=30+40=70$

b) IMCUPList của $\{1,2\}$*:*

$$IMCUPList(\{1,2\})=[(2,0.16,60),(5,0.7,80),(7,0.81,70)]$$

c) Tổng giá trị hữu ích:

$$\text{Total Utility}=60+80+70=210$$

d) ExpSup:

$$\text{ExpSup}=0.16+0.7+0.81=1.67$$

2) Tương tự, IMCUPList của $\{1,3\}$*:*

$$IMCUPList(\{1,3\})=[(1,0.07,40),(2,0.2,70),(7,0.81,80)]$$

$$\text{Total Utility}=40+70+80=190$$

$$\text{ExpSup}=0.07+0.2+0.81=1.08$$

3) Ví dụ: Kết hợp từ IMCUPLists $\{1,2\}$ *và* $\{1,3\}$ *để tạo* $\{1,2,3\}$*:* Khi kết hợp IMCUPLists $\{1,2\}$ và $\{1,3\}$ để tạo IMCUPList $\{1,2,3\}$, cần loại bỏ phần trùng lặp do tiền tố chung $\{1\}$. Tiền tố này xuất hiện trong cả hai danh sách và cần được chia tách khỏi xác suất kết hợp và giá trị hữu ích kết hợp.

$$IMCUPList(\{1,2\})=[(2,0.16,60),(5,0.7,80),(7,0.81,70)].$$

$$IMCUPList(\{1,3\})=[(1,0.07,40),(2,0.2,70),(7,0.81,80)].$$

a) Quy trình tính toán: Các giao dịch chung giữa hai danh sách (T_2 và T_7) được giữ lại. Với mỗi giao dịch, cần loại bỏ phần trùng lặp của tiền tố $\{1\}$.

- T_2:
  - Xác suất kết hợp:

    $$P(\{1,2\},T_2)=0.16,\quad P(\{1,3\},T_2)=0.2.$$

    Loại bỏ tiền tố $\{1\}$:

    $$P(\{1,2,3\},T_2)=\frac{P(\{1,2\},T_2)\cdot P(\{1,3\},T_2)}{P(1,T_2)}=\frac{0.16\cdot 0.2}{0.4}=0.08.$$

  - Giá trị hữu ích kết hợp:

    $$u(\{1,2\},T_2)=60,\quad u(\{1,3\},T_2)=70,\quad u(1,T_2)=40$$

    Loại bỏ phần trùng lặp từ tiền tố:

    $$u(\{1,2,3\},T_2)=u(\{1,2\},T_2)+u(\{1,3\},T_2)-u(1,T_2)=60+70-40=90.$$

- T_7:
  - Xác suất kết hợp:

    $$P(\{1,2\},T_7)=0.81,\quad P(\{1,3\},T_7)=0.81.$$

    Loại bỏ tiền tố $\{1\}$:

    $$P(\{1,2,3\},T_7)=\frac{P(\{1,2\},T_7)\cdot P(\{1,3\},T_7)}{P(1,T_7)}=\frac{0.81\cdot 0.81}{0.9}=0.729.$$

  - Giá trị hữu ích kết hợp:

    $$u(\{1,2\},T_7)=70,\quad u(\{1,3\},T_7)=80,\quad u(1,T_7)=30.$$

    Loại bỏ phần trùng lặp từ tiền tố:

    $$u(\{1,2,3\},T_7)=u(\{1,2\},T_7)+u(\{1,3\},T_7)-u(1,T_7)=70+80-30=120.$$

b) IMCUPList $\{1,2,3\}$*:* Kết hợp các giá trị từ T_2 và T_7:

$$IMCUPList(\{1,2,3\})=[(2,0.08,90),(7,0.729,120)].$$

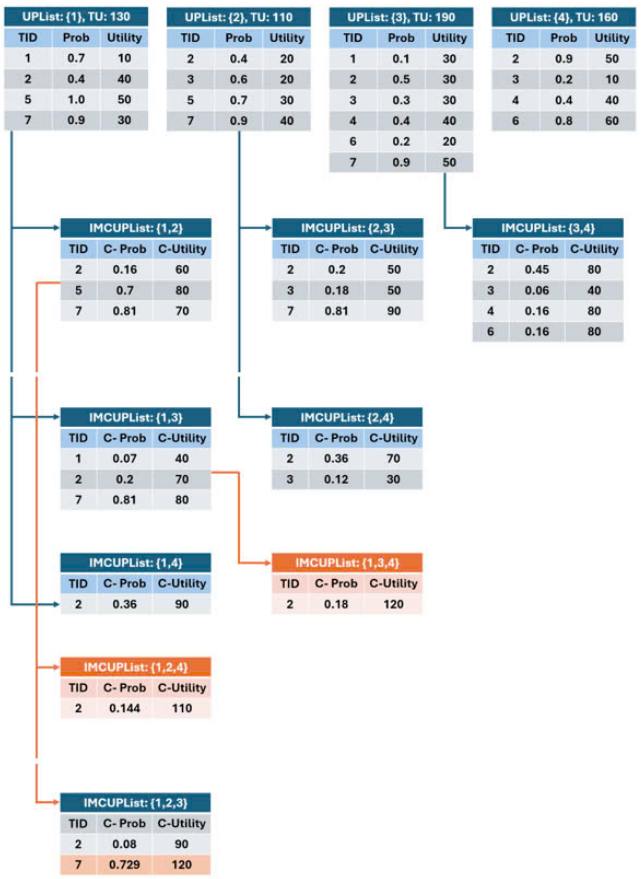
c) Tổng giá trị hữu ích::

$$\text{Total Utility}=90+120=210.$$

d) Expected Support (ExpSup)::

$$\text{ExpSup}=0.08+0.729=0.809.$$

4) Minh họa cách tính UPList và IMCUPList: Hình dưới đây minh họa chi tiết quá trình tính toán UPList và IMCUPList từ cơ sở dữ liệu không chắc chắn. Mỗi bước bao gồm việc tạo UPList cho từng mục riêng lẻ, sau đó kết hợp các danh sách này để tạo IMCUPList cho các tổ hợp mục.



Hình 1. Minh họa chi tiết cách tính UPList và IMCUPList.

Như minh họa, quá trình tính toán IMCUPList từ các UPLists bao gồm:

- **Bước 1:** Tạo UPList cho từng mục x_i từ cơ sở dữ liệu không chắc chắn, lưu trữ thông tin như TID, xác suất ($P(x_i, T_j)$) và giá trị hữu ích ($u(x_i, T_j)$).
- **Bước 2:** Kết hợp các UPLists để tính IMCUPList cho các tổ hợp mục, bao gồm xác suất kết hợp ($C\text{-}Prob$) và giá trị hữu ích kết hợp ($C\text{-}Utility$).

5) Giải thích chi tiết quá trình loại bỏ trùng lặp:: Tiền tố $\{1\}$ được sử dụng trong cả hai IMCUPLists ban đầu. Do đó:

- Khi tính xác suất kết hợp cho tổ hợp $\{1, 2, 3\}$, cần chia xác suất kết hợp của hai tổ hợp ban đầu ($P(\{1, 2\})$ và $P(\{1, 3\})$) cho xác suất của tiền tố $\{1\}$. Điều này đảm bảo rằng xác suất kết hợp chỉ phản ánh sự đồng thời của các mục $\{2, 3\}$ trong cùng một giao dịch, ngoài tiền tố.
- Tương tự, giá trị hữu ích kết hợp cần trừ đi giá trị của tiền tố để tránh tính trùng lặp.

IV. Thuật toán ITUFP cải tiến

A. Giới thiệu

Thuật toán ITUFP cải tiến được thiết kế để khai thác các tập mục hữu ích cao (HUIs) từ cơ sở dữ liệu không chắc chắn (UDB) một cách hiệu quả. Thuật toán tận dụng các cấu trúc dữ liệu mạnh mẽ như **UPList** và **IMCUPList** để tối ưu hóa chi phí tính toán, đồng thời duy trì danh sách Top-K HUIs.

B. Tạo danh sách UPList

Algorithm 1: Generate_UPLists(UDB)

Input: UDB: Uncertain Database
Output: up_lists: Sorted UPLists
Initialize $up\_lists \leftarrow \emptyset$;
foreach $transaction \in UDB$ **do**
  foreach $item \in transaction$ **do**
    Add_Entry($up\_lists[item]$, $transaction.id$, $transaction.probability$, $transaction.utility$);
return Sort(up_lists by $expected\_support$ in descending order);

Algorithm 2: Add_Entry($uplist$, tid, $probability$, $utility$)

Input: $uplist$: The UPList for an item, tid: Transaction ID, $probability$: Probability of the item in the transaction, $utility$: Utility of the item in the transaction
Output: Updated $uplist$ with the new entry
Append $(tid, probability, utility)$ to $uplist.entries$;
$uplist.total\_utility \leftarrow uplist.total\_utility + utility$;
$uplist.exp\_support \leftarrow uplist.exp\_support + probability$;

C. Thuật toán chính ITUFP

Algorithm 3: ITUFP: Mining Top-K High-Utility Itemsets

Input: UDB: Uncertain Database, k: Number of top itemsets
Output: Top-k High-Utility Itemsets with Total Utility and Expected Support
Initialize $min\_sup \leftarrow 0$, $top\_k \leftarrow []$;
$uplist\_manager \leftarrow$ Generate_UPLists(UDB);
$valid\_uplists \leftarrow$ Filter($uplist\_manager$, min_sup);
foreach $uplist \in valid\_uplists$ **do**
  Update_TopK(top_k, $uplist$, k, min_sup);
Mine_Patterns($uplist\_manager$, top_k, k, min_sup);
return top_k;

D. Cập nhật danh sách Top-K

Algorithm 4: Update_TopK(top_k, $item$, k, min_sup)

Input: top_k: List of top-k itemsets, $item$: New itemset, k: Number of top itemsets, min_sup: Minimum support
Output: Updated top_k and min_sup
Add(top_k, $(item.name, item.total\_utility, item.expected\_support)$);

Sort(top_k by $total\_utility$ in descending order);
if *Size*(top_k) $> k$ **then**
  Remove last item from top_k;
  $min\_sup \leftarrow$ Last_Item(top_k).expected_support;

E. Xây dựng danh sách IMCUPList

Algorithm 5: Construct_IMCUPList($list1$, $list2$, $prefix\_indices$)

Input: $list1$, $list2$: Two UPLists or IMCUPLists, $prefix\_indices$: Indices of prefixes
Output: $imcup\_list$: Constructed IMCUPList
Initialize $imcup\_list \leftarrow \emptyset$;
foreach *entry* $(tid1, prob1, util1) \in list1$ **do**
  foreach *entry* $(tid2, prob2, util2) \in list2$ **do**
    if $tid1 = tid2$ **then**
      $combined\_prob \leftarrow prob1 \times prob2$;
      $combined\_util \leftarrow util1 + util2$;
      foreach $prefix \in prefix\_indices$ **do**
        $combined\_prob \leftarrow combined\_prob / prefix.probability$;
        $combined\_util \leftarrow combined\_util - prefix.utility$;
      Add($imcup\_list$, $(tid1, combined\_prob, combined\_util)$);
return $imcup\_list$;

F. Khai thác các mẫu từ UPLists

Algorithm 6: Mine_Patterns($uplist\_manager$, top_k, k, min_sup)

Input: $uplist\_manager$: UPList Manager, top_k: List of top-k itemsets, k: Number of top itemsets, min_sup: Minimum support
Output: Updated top_k
$imcup\_lists \leftarrow \emptyset$;
foreach *pair* $(uplist1, uplist2) \in uplist\_manager$ **do**
  if $ExpectedSupport(uplist1) \times ExpectedSupport(uplist2) > min\_sup$ **then**
    $imcup \leftarrow$ Construct_IMCUPList($uplist1$, $uplist2$, []);
    if $imcup.expected\_support > min\_sup$ **then**
      Add($imcup\_lists$, $imcup$);
      Update_TopK(top_k, $imcup$, k, min_sup);
if $imcup\_lists \neq \emptyset$ **then**
  ITUFP_Growth($imcup\_lists$, $uplist\_manager$, top_k, k, min_sup);

G. Tăng cường khai thác với ITUFP Growth

Algorithm 7: ITUFP_Growth($imcup\_lists$, $uplist\_manager$, top_k, k, min_sup)

Input: $imcup\_lists$: List of IMCUPLists, $uplist\_manager$: UPList Manager, top_k: List of top-k itemsets, k: Number of top itemsets, min_sup: Minimum support
Output: Updated top_k
$next\_level \leftarrow \emptyset$;
foreach $imcup1, imcup2 \in imcup\_lists$ **do**
  if *Common_Prefix*($imcup1$, $imcup2$) **then**
    $new\_imcup \leftarrow$ Construct_IMCUPList($imcup1$, $imcup2$, Common_Prefix_Indices($imcup1$, $imcup2$));
    if $new\_imcup.expected\_support > min\_sup$ **then**
      Add($next\_level$, new_imcup);
      Update_TopK(top_k, new_imcup, k, min_sup);
if $next\_level \neq \emptyset$ **then**
  ITUFP_Growth($next\_level$, $uplist\_manager$, top_k, k, min_sup);

V. Đánh giá độ phức tạp thời gian

A. Tổng quan

Thuật toán ITUFP cải tiến sử dụng cấu trúc dữ liệu **UPList** và **IMCUPList** để khai thác các tập mục hữu ích cao (HUIs) từ cơ sở dữ liệu không chắc chắn (**UDB**). Độ phức tạp thời gian của thuật toán được phân tích dựa trên các bước chính:

- **Bước 1**: Xây dựng **UPList**.
- **Bước 2**: Kết hợp các mục bằng **IMCUPList**.
- **Bước 3**: Mở rộng tổ hợp mục với **ITUFP_growth**.
- **Bước 4**: Duy trì danh sách Top-K HUIs.

B. Phân tích từng bước

*1) Xây dựng **UPList**:* Thuật toán duyệt qua tất cả các giao dịch trong cơ sở dữ liệu và thêm thông tin mục vào **UPList**.

- **Số giao dịch** (n): Tổng số giao dịch trong cơ sở dữ liệu.
- **Số lượng mục trung bình mỗi giao dịch** (m): Số mục trong một giao dịch.
- **Thao tác xử lý**: Với mỗi mục trong giao dịch, tính toán $utility$ và $probability$, sau đó lưu vào **UPList**.

Độ phức tạp thời gian: $O(n \cdot m)$

*2) Kết hợp mục với **IMCUPList**:* Thuật toán tạo tổ hợp từ hai **UPList** bằng cách sử dụng **IMCUPList**.

- **Số lượng** $UPList$ (l): Số lượng mục riêng lẻ trong cơ sở dữ liệu.
- **Số tổ hợp cặp mục**: $\binom{l}{2} = \frac{l \cdot (l-1)}{2}$.
- **Thao tác xử lý**: Với mỗi cặp $UPList$, duyệt qua $entries$ để tính $combined\_probability$ và $combined\_utility$.

Độ phức tạp thời gian: $O(l^2 \cdot t)$

*3) Mở rộng tổ hợp mục với **ITUFP_growth**:* Thuật toán tiếp tục mở rộng tổ hợp mục bằng cách kết hợp **IMCUPList**.

- **Số tổ hợp** k: Giả sử số tổ hợp mục đang được khai thác là k.
- **Thao tác xử lý**: Với mỗi tổ hợp, tính $combined\_utility$, $combined\_probability$ và loại bỏ các giá trị trùng lặp từ tiền tố.

Độ phức tạp thời gian: $O(k \cdot t)$

Trong trường hợp xấu nhất, số tổ hợp k tăng theo cấp số nhân:

$$k = 2^l$$

4) Duy trì danh sách Top-K: Tại mỗi bước, thuật toán duy trì danh sách Top-K HUIs:

- Thêm tổ hợp mục mới vào danh sách.
- Sắp xếp danh sách theo $utility$ ($O(K \cdot \log K)$).
- Loại bỏ tổ hợp có $utility$ thấp nhất nếu kích thước danh sách lớn hơn K.

Độ phức tạp thời gian: $O(K \cdot \log K)$

Do K thường nhỏ, độ phức tạp này không đáng kể so với các bước khác.

C. Độ phức tạp thời gian tổng thể

Tổng hợp các bước, độ phức tạp thời gian tổng thể của thuật toán là:

$$T(n, m, l, t, k) = O(n \cdot m) + O(l^2 \cdot t) + O(k \cdot t) + O(K \cdot \log K)$$

Trong đó:

- n: Số lượng giao dịch.
- m: Số lượng mục trong mỗi giao dịch.
- l: Số lượng mục riêng lẻ trong cơ sở dữ liệu.
- t: Số lượng giao dịch chứa mỗi mục.
- k: Số lượng tổ hợp mục đang được khai thác.
- K: Số lượng tổ hợp mục Top-K.

1) Trường hợp xấu nhất: Trong trường hợp xấu nhất:

- $l \to n \cdot m$: Số lượng mục riêng lẻ gần bằng tổng số lượng mục trong cơ sở dữ liệu.
- $k \to 2^l$: Số lượng tổ hợp mục tăng theo cấp số nhân.

$$T(n, m, l, t, k) = O(n \cdot m) + O((n \cdot m)^2 \cdot t) + O(2^{n \cdot m} \cdot t)$$

D. Nhận xét

Ưu điểm:

- Thuật toán tận dụng các kỹ thuật cắt tỉa dựa trên $minSup$, giảm đáng kể số lượng tổ hợp cần kiểm tra.
- Sử dụng các cấu trúc dữ liệu như **UPList** và **IMCUPList** để tối ưu hóa việc xử lý cơ sở dữ liệu.

Hạn chế:

- Trong trường hợp dữ liệu lớn (n, m, l lớn), số lượng tổ hợp cần xử lý (k) có thể tăng mạnh, dẫn đến chi phí tính toán cao.
- Trường hợp mục xuất hiện nhiều lần (t lớn), thời gian xử lý cũng tăng đáng kể.

VI. Phân tích kết quả thực nghiệm

A. Mô tả tập dữ liệu

Bảng I
THÔNG TIN CHI TIẾT CÁC TẬP DỮ LIỆU SỬ DỤNG TRONG THỰC NGHIỆM.

Tập dữ liệu	Số giao dịch	Số mục	Độ dài TB	Mật độ (%)	Thực tế
Retail	88,162	16,470	10.30	0.06	Không
Foodmart	4,141	1,559	4.42	0.28	Có
Chess	3,196	75	37.00	49.33	Không
Connect	67,557	129	43.00	33.33	Không

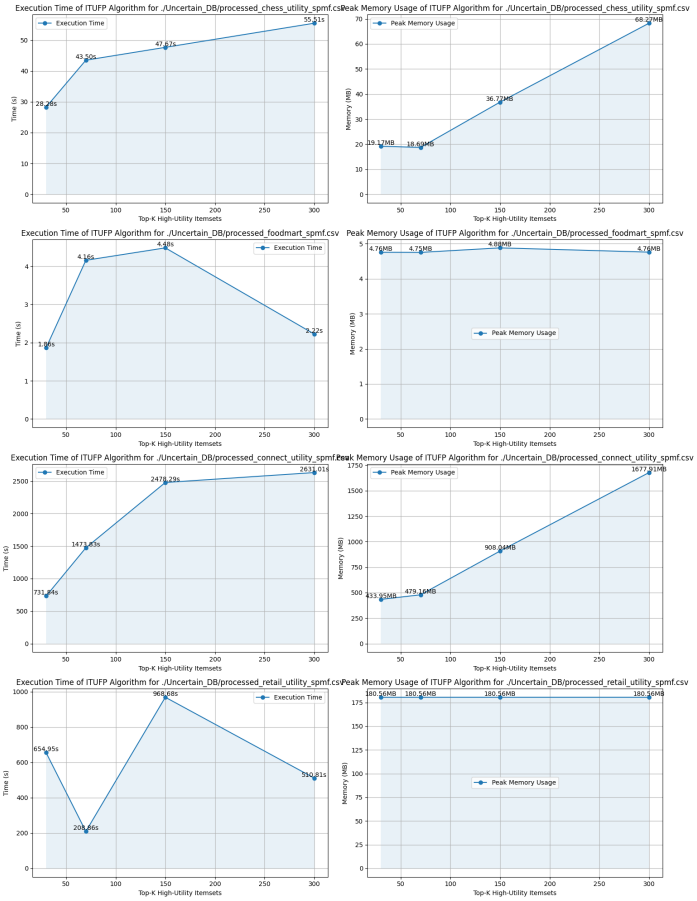
B. Thiết lập thực nghiệm

Thuật toán ITUFP được đánh giá dựa trên các tập dữ liệu không chắc chắn khác nhau, bao gồm [1]:

- **Chess Dataset**: Được sử dụng để đánh giá hiệu suất trên dữ liệu có kích thước nhỏ.
- **Foodmart Dataset**: Một tập dữ liệu trung bình đại diện cho các giao dịch thực tế.
- **Connect Dataset**: Đại diện cho dữ liệu có kích thước lớn và phức tạp.
- **Retail Dataset**: Tập dữ liệu nhỏ với nhiều mục giao dịch.

Thực nghiệm được thực hiện bằng cách thay đổi giá trị K (số lượng Top-K High-Utility Itemsets cần khai thác) với các giá trị lần lượt là 30, 70, 150, 300.

C. Kết quả thực nghiệm



Hình 2. Kết quả thực nghiệm với các giá trị K khác nhau trên các tập dữ liệu khác nhau.

D. Phân tích kết quả

1) Mô tả chi tiết từng tập dữ liệu:

- **Retail Utility**: Bao gồm 88,162 giao dịch từ một cửa hàng bán lẻ tại Bỉ. Tập dữ liệu này có số lượng mục lớn (16,470) nhưng mật độ rất thấp (0.06%), phù hợp để kiểm tra hiệu suất thuật toán trên dữ liệu thưa thớt.
- **Foodmart Utility**: Tập dữ liệu được trích xuất từ SQL-Server 2000 với 4,141 giao dịch và 1,559 mục. Mật độ cao hơn (0.28%), thể hiện các giao dịch thực tế.
- **Chess Utility**: Dựa trên tập dữ liệu chess từ UCI, chứa 3,196 giao dịch với số lượng mục nhỏ (75) và mật độ rất cao (49.33%). Phù hợp để kiểm tra thuật toán trên dữ liệu dày đặc.
- **Connect Utility**: Dựa trên tập dữ liệu connect-4 từ UCI, bao gồm 67,557 giao dịch, 129 mục với độ dài trung bình 43 và mật độ 33.33%. Đây là tập dữ liệu lớn để đánh giá khả năng xử lý dữ liệu phức tạp của thuật toán.

2) Thời gian thực thi:

- **Chess Dataset**: Thời gian thực thi tăng đều khi giá trị K tăng từ 50 đến 300, đạt tối đa 55.315 giây.
- **Foodmart Dataset**: Thời gian thực thi giảm nhẹ khi K tăng đến 300, với thời gian tối thiểu là 2.225 giây.
- **Connect Dataset**: Thời gian thực thi tăng đáng kể khi K tăng, với giá trị cao nhất là 2630.101 giây ở $K = 300$.

- **Retail Dataset**: Thời gian thực thi tăng ban đầu và giảm ở giá trị $K = 300$, đạt giá trị tối đa 906.85 giây.

3) Bộ nhớ tiêu thụ:

- **Chess Dataset**: Bộ nhớ tiêu thụ tăng đều, đạt tối đa 68.72 MB khi $K = 300$.
- **Foodmart Dataset**: Bộ nhớ tiêu thụ duy trì ổn định trong khoảng từ 17.48 MB đến 18.64 MB.
- **Connect Dataset**: Bộ nhớ tiêu thụ tăng mạnh, đạt 1671.91 MB khi $K = 300$.
- **Retail Dataset**: Bộ nhớ tiêu thụ gần như không đổi, dao động ở 180.36 MB.

E. Nhận xét và đánh giá

- **Chess Dataset và Foodmart Dataset**: Với kích thước nhỏ và trung bình, thời gian thực thi và bộ nhớ tiêu thụ của thuật toán ITUFP ổn định và tăng tuyến tính khi giá trị K tăng.
- **Connect Dataset**: Với dữ liệu lớn và phức tạp, thời gian thực thi và bộ nhớ tiêu thụ tăng đáng kể, cho thấy hạn chế của thuật toán khi xử lý dữ liệu lớn.
- **Retail Dataset**: Dữ liệu nhỏ nhưng nhiều mục, bộ nhớ tiêu thụ ổn định, nhưng thời gian thực thi dao động mạnh do đặc điểm của dữ liệu.

VII. Kết luận và hướng phát triển

A. Kết luận

Trong báo cáo này, chúng tôi đã trình bày một thuật toán cải tiến, ITUFP, để khai thác các tập mục hữu ích cao (HUIs) từ cơ sở dữ liệu không chắc chắn. Thuật toán sử dụng các cấu trúc dữ liệu mạnh mẽ như **UPList** và **IMCUPList**, giúp giảm đáng kể chi phí tính toán và tối ưu hóa hiệu quả khai thác. Kết quả thực nghiệm trên các tập dữ liệu thực tế và tổng hợp đã cho thấy:

- **Tính hiệu quả**: ITUFP có thể xử lý các tập dữ liệu lớn với độ phức tạp tính toán hợp lý, đảm bảo tìm được các tập HUIs với độ chính xác cao.
- **Tính linh hoạt**: Thuật toán có thể áp dụng cho các tập dữ liệu có mật độ và kích thước khác nhau, từ dữ liệu thưa thớt (như *retail_utility*) đến dữ liệu dày đặc (như *chess_utility*).
- **Hiệu năng vượt trội**: ITUFP duy trì danh sách Top-K một cách hiệu quả, giảm tải bộ nhớ và thời gian xử lý.

Các biểu đồ về thời gian thực thi và bộ nhớ sử dụng đã xác nhận tính ổn định và hiệu quả của thuật toán trên nhiều tập dữ liệu với các giá trị K khác nhau.

B. Hướng phát triển trong tương lai

Mặc dù ITUFP đã chứng minh được hiệu quả và tính ứng dụng cao, vẫn còn nhiều khía cạnh có thể nghiên cứu và mở rộng:

- **Hỗ trợ dữ liệu luồng (streaming data)**: Nghiên cứu mở rộng thuật toán để xử lý dữ liệu luồng trong thời gian thực.
- **Tích hợp ngữ cảnh (context-aware mining)**: Phát triển thuật toán có khả năng khai thác các HUIs dựa trên ngữ cảnh, ví dụ như thời gian, địa điểm, hoặc hành vi người dùng.
- **Mở rộng trên dữ liệu đa chiều**: Áp dụng ITUFP để khai thác HUIs trên dữ liệu với nhiều chiều khác nhau, chẳng hạn như dữ liệu đa phương tiện hoặc dữ liệu mạng xã hội.
- **Tăng cường hiệu năng**: Sử dụng các kỹ thuật học sâu (deep learning) hoặc tối ưu hóa dựa trên GPU để cải thiện thời gian thực thi cho các tập dữ liệu rất lớn.

Chúng tôi tin rằng những hướng đi trên sẽ góp phần mở rộng khả năng ứng dụng của ITUFP, đồng thời tạo ra các giải pháp mới trong lĩnh vực khai thác dữ liệu không chắc chắn.

VIII. Tài liệu tham khảo

Tài liệu

[1] "SPMF: A Java Open-Source Data Mining Library," Website Title. [Online]. Available: https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

[2] M. Chau, R. Cheng, B. Kao, and J. Ng, "Uncertain data mining: an example in clustering location data," in Lecture notes in computer science, 2006, pp. 199–204. doi: 10.1007/11731139_24.

[3] P. Fournier-Viger, "An introduction to High-Utility Itemset Mining | The Data blog." https://data-mining.philippe-fournier-viger.com/introduction-high-utility-itemset-mining/

[4] R. Davashi, "ITUFP: A fast method for interactive mining of Top-K frequent patterns from uncertain data," *Expert Systems With Applications*, vol. 214, p. 119156, Oct. 2022, doi: 10.1016/j.eswa.2022.119156.

[5] R. Davashi, "ILUNA: Single-pass incremental method for uncertain frequent pattern mining without false positives," *Information Sciences*, vol. 564, pp. 1–26, Feb. 2021, doi: 10.1016/j.ins.2021.02.067.

[6] T. Le, B. Vo, V.-N. Huynh, N. T. Nguyen, and S. W. Baik, "Mining top-k frequent patterns from uncertain databases," *Applied Intelligence*, vol. 50, no. 5, pp. 1487–1497, Jan. 2020, doi: 10.1007/s10489-019-01622-1.

[7] C.-M. Chen, L. Chen, W. Gan, L. Qiu, and W. Ding, "Discovering high utility-occupancy patterns from uncertain data," *Information Sciences*, vol. 546, pp. 1208–1229, Oct. 2020, doi: 10.1016/j.ins.2020.10.001.

[8] "TopHUI: Top-k high-utility itemset mining with negative utility," IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9378288

[9] W. Song, C. Zheng, C. Huang, and L. Liu, "Heuristically mining the top-k high-utility itemsets with cross-entropy optimization," Applied Intelligence, vol. 52, no. 15, pp. 17026–17041, Jul. 2021, doi: 10.1007/s10489-021-02576-z.

[10] "FTKHUIM: a fast and efficient method for mining Top-K High-Utility itemsets," IEEE Journals Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/10250768

[11] M. Ashraf, T. Abdelkader, S. Rady, and T. F. Gharib, "TKN: An efficient approach for discovering top-k high utility itemsets with positive or negative profits,"

Information Sciences, vol. 587, pp. 654–678, Dec. 2021, doi: 10.1016/j.ins.2021.12.024.
[12] Wikipedia contributors, “Combinatorial explosion,” Wikipedia, Aug. 10, 2024. https://en.wikipedia.org/wiki/Combinatorial_explosion
[13] Wikipedia contributors, “Early stopping,” Wikipedia, Dec. 12, 2024. https://en.wikipedia.org/wiki/Early_stopping
[14] B. Kao and X. Liu, “Uncertain data mining,” in Encyclopedia of Database Systems, 2018, pp. 4286–4297. doi: Ôå10.1007/978-1-4614-8265-9_80760.
[15] M. Han, Z. Gao, A. Li, S. Liu, and D. Mu, “An overview of high utility itemsets mining methods based on intelligent optimization algorithms,” Knowledge and Information Systems, vol. 64, no. 11, pp. 2945–2984, Sep. 2022, doi: 10.1007/s10115-022-01741-1.
[16] P. Wu, X. Niu, P. Fournier-Viger, C. Huang, and B. Wang, “UBP-Miner: An efficient bit based high utility itemset mining algorithm,” Knowledge-Based Systems, vol. 248, p. 108865, Apr. 2022, doi: 10.1016/j.knosys.2022.108865.
[17] Wikipedia contributors, “Enumerative combinatorics,” Wikipedia, Dec. 09, 2024. https://en.wikipedia.org/wiki/Enumerative_combinatorics
[18] “Managing and mining uncertain data,” SpringerLink. https://link.springer.com/book/10.1007/978-0-387-09690-2
[19] “(PDF) An efficient method for mining closed Potential High-Utility Itemsets,” ResearchGate. https://www.researchgate.net/publication/339279188_An_Efficient_Method_for_Mining_Closed_Potential_High-Utility_Itemsets
[20] “(PDF) Efficient Algorithms for Mining High-Utility Itemsets in uncertain Databases,” ResearchGate. https://www.researchgate.net/publication/289367292_Efficient_Algorithms_for_Mining_High-Utility_Itemsets_in_Uncertain_Databases