Thuật toán khai thác tiện ích bảo vệ quyền riêng tư nhanh chóng dựa trên từ điển danh sách tiện ích

Chunyong Yin [1] ⊙ **Ying Li** [1]

Accepted: 12 June 2023 / Published online: 27 October 2023

Tóm tắt

Khai thác tiện ích bảo vệ quyền riêng tư (PPUM) nhằm giải quyết vấn đề rò rỉ thông tin nhạy cảm trong khai thác mẫu tiện ích. Trong những năm gần đây, các nhà nghiên cứu đã đề xuất các thuật toán để giải quyết vấn đề bảo vệ quyền riêng tư. Tuy nhiên, các thuật toán này có tác dụng phụ cao, thời gian chuẩn hóa lâu và độ phức tạp tính toán. Mặc dù thuật toán FPUTT giảm số lần quét cơ sở dữ liệu nhưng việc xây dựng cây và truyền tải vẫn mất nhiều thời gian. Bài báo đề xuất thuật toán từ điển danh sách tiện ích nhanh (FULD). Từ điển danh sách tiện ích bao gồm tất cả các mục nhạy cảm. Thông qua tra cứu từ điển, các mục nhạy cảm có thể được tìm thấy và sửa đổi. Ngoài ra, các khái niệm mới về SINS và tns được đề xuất để giảm tác dụng phụ của thuật toán. Trong bài báo này, các thử nghiệm cho thấy thuật toán FULD có hiệu suất tốt như thời gian chạy và tác dụng phụ. Thời gian chạy của FULD ngắn hơn 15–20 lần so với thuật toán FPUTT. Nó hoạt động tốt cả trên các tập dữ liệu thưa thớt và dày đặc.

Keywords Sensitive information · Utility mining · Privacy preserving · Utility-list dictionary · Utility lists

1 Giới thiệu

Khai thác dữ liệu [1] nhằm mục đích trích xuất thông tin hữu ích từ dữ liệu có cấu trúc hoặc không có cấu trúc và cấu trúc nó để mọi người sử dụng. Trong những năm gần đây, khai thác dữ liệu [2–5] có thể đáp ứng các ứng dụng thực tế trong nhiều lĩnh vực khác nhau [1, 6–8]. Mặc dù các kỹ thuật này rất phổ biến nhưng các thuật toán này bỏ qua tiện ích của các tập mục như tầm quan trọng, sự quan tâm, sự hài lòng hoặc rủi ro. Nói chung, những thông tin tiềm ẩn về tiện ích là điều bình thường trong cuộc sống.Các thuật toán khai thác dữ liệu truyền thống thường không thể khai thác được thông tin thực sự quan trọng đối với người dùng. Do đó, thuật toán Khai thác mẫu tiện ích cao (HUPM) [9] đã xuất hiện, thuật toán này có thể thiết lập tiện ích của một mục theo tầm quan trọng của nó.

Tiện ích là thước đo chủ quan phản ánh mong muốn của con người ở một mức độ nào đó. Trong Wikipedia, tiện ích đề cập đến mức độ hài lòng của người dùng đối với hàng hóa hoặc dịch vụ. Trong [10], các tác giả đã tóm tắt một số biện pháp. Chúng là các thước đo khách quan, các thước đo chủ quan và các thước đo dựa trên ngữ nghĩa [10–12]. Các thước đo khách quan [12, 13] được đo lường theo số liệu thống kê và có tiêu chuẩn khắt khe, chẳng hạn như tiện ích tập mục trong khai thác tiện ích. Các thước đo chủ quan và ngữ nghĩa [9, 14] được xác định dựa trên kiến thức nền tảng của người dùng. Chúng phù hợp cho người dùng có kinh nghiệm. Do kiến thức nền tảng khác nhau của người dùng nên các phương pháp dựa trên ngữ nghĩa cũng khác nhau. Trong khai thác tiện ích, mọi người sử dụng tiện ích để đo lường tầm quan trọng của một vật phẩm.

Khai thác mẫu tiện ích cao (HUPM) có thể khai thác thông tin quan trọng. Nhưng nó có một lỗ hổng trong việc bảo vệ quyền riêng tư. Do đó, các học giả đã đề xuất Khai thác tiện ích bảo vệ quyền riêng tư (PPUM). Nghiên cứu trong lĩnh vực này vẫn còn ở giai đoạn sơ khai. Các phương pháp bảo vệ quyền riêng tư là quyền riêng tư khác biệt, công nghệ mã hóa và chuỗi khối.Quyền riêng tư khác biệt [15] có thể đảm bảo sự cân bằng giữa quyền riêng tư và hiệu quả. Công nghệ mã hóa [16] mã hóa dữ liệu gốc thành văn bản mã hóa để đảm bảo tính riêng tư của dữ liệu trong quá trình lưu trữ và truyền tải. Chuỗi khối [17] có thể giám sát các hành vi bất thường của thiết bị hoặc máy chủ. Nhưng thông lượng của nó bị hạn chế và khả năng mở rộng cũng kém. Tuy nhiên, những kỹ thuật này không được áp dụng trong PPUM, điều này sẽ dẫn đến tổn thất nghiêm trọng về tiện ích.

Để bảo vệ thông tin nhạy cảm trong khai thác tiện ích, các nhà nghiên cứu áp dụng các phương pháp giảm tiện ích nội bộ, chèn hoặc xóa giao dịch.Yeh và cộng sự.[18] lần đầu tiên đề xuất chiến lược bảo vệ quyền riêng tư trong lĩnh vực này. Thuật toán loại bỏ một số tập mục nhạy cảm để cơ sở dữ liệu đã được làm sạch không còn chứa thông tin nhạy cảm. Gan và cộng sự.[19] đề xuất MSU-MAU và MSU MIU, áp dụng cơ chế chiếu để tăng tốc quá trình dọn dẹp cơ sở dữ liệu gốc. Tuy nhiên, các thuật toán này mất nhiều thời gian để chạy trên các tập dữ liệu lớn. Ngoài các phương pháp trên, Yun et al.[20] đã xây dựng cấu trúc cây để lưu trữ thông tin mục, giảm số lần quét cơ sở dữ liệu. Tuy nhiên, thuật toán có tác dụng phụ lớn và độ phức tạp tính toán cao.Li và cộng sự.[21] đề xuất một thuật toán chính xác dựa trên quy hoạch tuyến tính số nguyên. Mặc dù thuật toán này có thể thu được lời giải chính xác nhưng nó có độ phức tạp tính toán cao và mất nhiều thời gian. Vì vậy, để giải quyết các vấn đề này, bài báo đề xuất thuật toán FULD. Những đóng góp chính của bài viết này được tóm tắt như sau:

- Trong các nghiên cứu trước đây về khai thác tiện ích bảo vệ quyền riêng tư, các nhà nghiên cứu đã sử dụng cấu trúc cây, thuật toán heuristic, v.v. để tăng tốc quá trình dọn dẹp. Lần đầu tiên, bài viết này đề xuất một cấu trúc từ điển danh sách tiện ích mới để giải quyết những vấn đề này. UTLDic được đề xuất trong bài viết này bao gồm các mục nhạy cảm và UTLists. Cấu trúc này hoàn toàn khác với cấu trúc danh sách tiện ích trong khai thác mẫu có tính tiện ích cao. Nó có thể giảm số lượng tính toán tiện ích và tìm kiếm thông tin nhạy cảm mà không cần quét cơ sở dữ liệu nhiều lần.
- Để giảm bớt tác dụng phụ của thuật toán, bài báo đề xuất các khái niệm SINS và tns. SINS phản ánh số lần xuất hiện của các mục nhạy cảm trong các tập mục có tính tiện ích cao không nhạy cảm. Tns thể hiện mối quan hệ giữa giao dịch và các tập mục hữu ích cao không nhạy cảm. Thông qua những khái niệm mới này, hiệu suất của thuật toán có thể được tối ưu hóa hơn nữa.
- Từ các thử nghiệm, có thể thấy rằng hiệu suất của FULD bị ảnh hưởng bởi nhiều yếu tố, bao gồm số lượng tập mục nhạy cảm, mật độ của tập dữ liệu, ngưỡng tiện ích tối thiểu, kích thước của tập dữ liệu, v.v.
- Các thí nghiệm được thực hiện trên nhiều bộ dữ liệu và được so sánh với các thuật toán khác. Các thử nghiệm cho thấy thuật toán FULD hoạt động tốt trên các tập dữ liệu thưa thớt và dày đặc. Ngoài ra, thuật toán FULD ngắn hơn thuật toán FPUTT từ 15–20 lần.

Phần còn lại của bài báo được tổ chức như sau. Trong Phần 2, chúng tôi xem xét các công việc liên quan về HUPM và PPUM. Chi tiết về kiến thức nền tảng của PPUM được mô tả trong Phần 3. Trong Phần 4, chúng tôi giới thiệu thuật toán FULD. Trong Phần 5, chúng tôi đã tiến hành các thí nghiệm so sánh sâu rộng trên bốn bộ dữ liệu. Trong Phần 6, chúng tôi phân tích độ phức tạp của thuật toán FULD. Cuối cùng, chúng tôi kết luận bài viết này và hướng nghiên cứu tiếp theo trong Phần 7.

2 Công việc liên quan

Khai thác mẫu có tính tiện ích cao có thể thu được thông tin quan trọng từ dữ liệu lớn. Khai thác tiện ích bảo vệ quyền riêng tư giải quyết các vấn đề về quyền riêng tư trong khai thác tiện ích. Để hiểu rõ hơn, công việc liên quan được trình bày trong phần này.

2.1 Khai thác mẫu có tính tiện ích cao

Với số lượng thiết bị IoT ngày càng tăng, việc khai thác thông tin trong thời gian ngắn là điều khó khăn. Vì vậy, Lin và cộng sự.[22] đã đề xuất thuật toán MCUI-Miner. Thuật toán kết hợp mô hình GA và MapReduce để tăng tốc quá trình tìm kiếm. Gan [23] và cộng sự.tóm tắt các thuật toán khai thác mẫu tiện ích. Ưu điểm và nhược điểm của thuật toán được phân tích bằng cách phân loại. Tác giả đã xem xét một số phần mềm và bộ dữ liệu nguồn mở. Lin và cộng sự.[24] đã đề xuất mô hình DFM-Miner. Mô hình có thể khai thác tập mục tiện ích cao khép kín trong môi trường song song và phân tán. Mô hình này có hiệu suất tốt hơn trong việc xử lý các cơ sở dữ liệu lớn với mức sử dụng ít bộ nhớ hơn. Lin và cộng sự.[25] đã sử dụng mô hình phân cụm để nhóm các giao dịch có mối tương quan cao. Các tập mục tiện ích cao đóng có thể được thu được bằng mô hình GA nhỏ gọn trong thời gian ngắn. Thuật toán TGA có hiệu suất tốt về độ chính xác và số lượng mô-đun nhận dạng.

Để khai thác thông tin từ cơ sở dữ liệu, nhiều thuật toán khai thác mẫu tiện ích được các học giả đề xuất. Trong [26], thuật toán EHMIN có thể xem xét không chỉ lợi nhuận đơn vị dương trong dự án mà còn cả lợi nhuận đơn vị âm. Thuật toán chủ yếu tập trung vào việc sử dụng lợi nhuận đơn vị âm của dự án để khai thác các mẫu hữu ích. Đồng thời, thuật toán hoạt động tốt ở tất cả các chỉ số. Lee và cộng sự.[27] đã cải thiện việc khai thác mẫu có thể xóa truyền thống bằng cách giới thiệu cấu trúc danh sách, thêm thuộc tính số lượng và giá của các mặt hàng để ước tính lợi nhuận và trích xuất mẫu có thể xóa được tối đa theo sở thích của người dùng. Ryu và cộng sự.[28] đã đề xuất phương pháp khai thác HUOMI, có thể được sử dụng để xử lý các luồng dữ liệu được tạo động, đồng thời phân tích dữ liệu lớn một cách hiệu quả và nhanh chóng trên cơ sở dữ liệu ngày càng tăng. Lee và cộng sự.[27] đã đề xuất một thuật toán hiệu quả để khai thác các mẫu tiện ích trung bình cao dựa trên cửa sổ trượt với cấu trúc danh sách, được sử dụng để khai thác các mẫu có giá trị mới nhất trong luồng dữ liệu.

Để lưu trữ các tập mục thường xuyên, thuật toán FP-Growth [5] được đề xuất. Nó nén thông tin trong cơ sở dữ liệu và giảm số lần quét cơ sở dữ liệu. Lấy cảm hứng từ thuật toán này, Hu et al.[29] đã đề xuất thuật toán khai thác tập mục thường xuyên HYP để xác định các kết hợp có tính tiện ích cao. Thuật

toán giảm việc khai thác dữ liệu thành một vấn đề tối ưu hóa và thu được các giải pháp gần đúng thông qua Cây phân vùng năng suất cao. Tương tự như cấu trúc cây FP, Lin et al.[30] đã trình bày thuật toán tăng trưởng HUP. Cây HUP không chỉ có thể lưu trữ thông tin khai thác mà còn giảm thời gian quét cơ sở dữ liệu. Nhưng khác với cây FP truyền thống, mỗi nút được gắn với một mảng để giữ số lượng các mục tiền tố của nó trong đường dẫn.

Để đạt được hiệu quả cao hơn so với khai thác mẫu tiện ích dựa trên cây, các cấu trúc dữ liệu mới được đề xuất để khai thác các tập mục, chẳng hạn như HUP-Miner [31], EFIM [32] và d2HUP [33]. Kim và cộng sự.[34] đề xuất một cải tiến dựa trên thuật toán hiệu quả cao của phương pháp cửa sổ trượt. Nó khai thác các mẫu tiện ích cao bằng cách liên tục chèn dữ liệu dòng suy giảm vào dữ liệu mới nhất. Baek và cộng sự.[35] đã đề xuất thuật toán Khai thác mẫu tiện ích cao gần đúng (AHUPM). Thuật toán đã xác định hệ số dung sai tiện ích để tính toán phạm vi tiện ích đáng tin cậy cho các mẫu và do đó trích xuất các mẫu tiện ích cao được xác định là các mẫu tiện ích cao gần đúng từ cơ sở dữ liệu.

2.2 Khai thác tiện ích bảo vệ quyền riêng tư

Mục tiêu của HUPM là khai thác các tập mục hữu ích cao (HUI) và khám phá những thông tin thú vị. Nhưng việc khai thác mẫu tiện ích có thể tiết lộ thông tin nhạy cảm. Theo đó, HUPM còn vướng phải vấn đề rò rỉ thông tin nhạy cảm. Mọi người thường cần ẩn các tập mục hữu ích cao (SHUI) nhạy cảm này trong các tình huống thực tế. Vì vậy, PPUM đã trở thành tâm điểm nghiên cứu của các học giả trong những năm gần đây.

Công dụng của một vật bằng tổng công dụng bên trong của nó nhân với công dụng bên ngoài của nó. Nói chung, các phương pháp làm giảm công dụng của một vật phẩm có thể tóm tắt như sau: 1) trực tiếp giảm công dụng bên trong của vật phẩm đó, 2) thay đổi công dụng bên ngoài của vật phẩm đó. Tuy nhiên, tiện ích bên ngoài thường cố định và sẽ không thay đổi trong thế giới thực.

Vì vậy, việc giảm bớt công dụng nội tại của món đồ sẽ hợp lý hơn. Yeh và cộng sự.[18] đã đề xuất thuật toán HHUIF và MSICF, không chỉ che giấu thông tin nhạy cảm mà còn giảm thiểu tác dụng phụ của thuật toán. Yun và cộng sự.[20] đã giới thiệu một cấu trúc cây mới để đẩy nhanh quá trình nhiễu loạn và giảm không gian tìm kiếm của PPUM. Lin và cộng sự.[19] đề xuất thuật toán MSU-MAU và MSU-MIU để giảm thiểu tác dụng phụ do ẩn các tập mục nhạy cảm. Trong [36], SIF-IDF là thuật toán tham lam có khả năng loại bỏ các thông tin nhạy cảm trong giao dịch. Thuật toán dựa trên khái niệm tần suất thuật ngữ và tần suất tài liệu nghịch đảo (TF-IDF) trong khai thác văn bản. Nó có thể ẩn các mục thích hợp sau khi tính toán độ tương tự giữa các mục trong giao dịch và các tập mục nhạy cảm mong muốn.

Ngoài những điều trên, Li et al.[21] đã đề xuất một thuật toán mới dựa trên lập trình tuyến tính số nguyên (ILP) để giảm các tác dụng phụ được tạo ra trong quá trình dọn dẹp, trong đó mô tả việc ẩn quy trình tập mục nhạy cảm là một vấn đề thỏa mãn các ràng buộc. Jangra và cộng sự.[37] đã đề xuất hai thuật toán dựa trên kinh nghiệm để ẩn SHUI. Thuật toán thiết kế hai phương pháp cho các tập mục nhạy cảm và cải thiện tiện ích dữ liệu bằng cách giảm thiểu chi phí bị thiếu. Trong [38], Khung khai thác bảo mật và bảo mật quyền riêng tư (PPSF) được đề xuất. Khung này cung cấp một số thuật toán PPUM, chẳng hạn như HHUIF, MSICF, v.v. Người dùng có thể thực thi thuật toán bảo vệ quyền riêng tư thông qua giao diện hoạt động. Wu và cộng sự.[39] đã sử dụng kỹ thuật ngưỡng động tối thiểu để ẩn thông tin nhạy cảm. Ngoài ra, thuật toán kết hợp với GA có thể giảm thiểu tác dụng phụ. So với các thuật toán truyền thống, thuật toán có thể che giấu những thông tin nhạy cảm hơn. Để giảm bớt tác dụng phụ hơn nữa, Lin et al.[40] đề xuất một phương pháp tối ưu hóa đàn kiến. Nó ẩn thông tin nhạy cảm thông qua việc xóa giao dịch. Mỗi con kiến trong quần thể được biểu thị là một nhóm các giao dịch xóa có thể xảy ra. So với thuật toán di truyền, phương pháp này có hiệu suất tốt hơn.

Tuy nhiên, nhiệm vụ quan trọng là ẩn HUI và mở rộng các thuật toán này để xử lý dữ liệu tuần tự. Trong [41], tác giả lần đầu tiên đề xuất thuật toán HHUSP và MSPCF để ẩn tất cả các tập mục nhạy cảm. Ngoài ra, Huynh và cộng sự.[42] đề xuất thuật toán HHUSP-A và HHUSP-D. Các thuật toán này giảm thời gian thực hiện thuật toán và chi phí mất mát theo thứ tự tăng dần hoặc giảm dần, từ đó cải thiện hiệu suất của HHUSP.

3 Kiến thức nền tảng

Để hiểu khai thác tiện ích bảo vệ quyền riêng tư (PPUM), kiến thức sơ bộ và vấn đề về PPUM được giới thiệu ngắn gọn trong phần này.

3.1 Định nghĩa

Cơ sở dữ liệu được ký hiệu là $D = \{T_1, T_2, \cdots, T_n\}$, bao gồm n transactions. Một transaction $T_k (1 \leq k \leq n)$ được tạo thành từ một tập hữu hạn các item. Cho $I = \{i_1, i_2, \cdots, i_m\}$ là m các item khác nhau và T_k là tập con của I. r-itemset có nghĩa là nó bao gồm r các item, chẳng hạn như 2-itemset $I = \{A, C\}$.

Định nghĩa 1 Tiện ích nội tại của $i_v (1 \leq v \leq m)$ trong T_n, ký hiệu là $q(i_v, T_n)$, là số lượng của i_v trong T_n. Ví dụ ở Bảng 1, $q(C, T_2) = 2$.

Định nghĩa 2 Tiện ích bên ngoài của i_v, ký hiệu là $p(i_v)$, thể hiện tầm quan trọng, lợi nhuận, số lần nhấp vào liên kết trang, v.v. Ví dụ trong Bảng 2, $p(C) = 4$.

Bảng 1: Cơ sở dữ liệu transaction

TID	AA	BB	CC	DD	EE	tu
T_1	0	0	7	1	1	41
T_2	1	0	2	0	2	31
T_3	0	6	4	3	7	149
T_4	0	5	3	9	0	121
T_5	3	0	10	3	0	85
T_6	0	0	5	0	9	83
T_7	6	0	9	2	5	137
T_8	1	6	2	5	3	134

Định nghĩa 3 Tiện ích của i_v trong T_n, ký hiệu là $u(i_v T_n)$, như sau:

$$u(i_v, T_n) = q(i_v, T_n) \times p(i_v) \tag{1}$$

Ví dụ trong Bảng 1, $u(C, T_2) = q(C, T_2) \times p(C) = 2 \times 4 = 8$.

Định nghĩa 4 Tiện ích của itemset X trong $T_n (X \subseteq T_n)$, ký hiệu là $u(X, T_n)$, được biểu diễn dưới dạng

$$u(X, T_n) = \sum_{i_v \in X} u(i_v, T_n) \tag{2}$$

Ví dụ trong Bảng 1, $u(AC, T_2) = u(A, T_2) + u(C, T_2) = 1 \times 9 + 2 \times 4 = 17$.

Định nghĩa 5 Tiện ích của itemset X trong cơ sở dữ liệu D, ký hiệu là $u(X)$, được trình bày dưới đây:

$$u(X) = \sum_{i_v \in X, X \subseteq T_n} u(X, T_n) \tag{3}$$

Ví dụ trong Bảng 1, $u(AC) = u(AC, T_2) + u(AC, T_5) + u(AC, T_7) + u(AC, T_8) = 1 \times 9 + 2 \times 4 + 3 \times 9 + 10 \times 4 + 6 \times 9 + 9 \times 4 + 1 \times 9 + 2 \times 4 = 191$.

Định nghĩa 6 Tiện ích của T_n trong cơ sở dữ liệu D, ký hiệu là $tu(T_n)$, được biểu diễn như sau:

$$tu(T_n) = \sum_{i_v T_n} u(i_v, T_n) \tag{4}$$

Ví dụ trong Bảng 1, $tu(T_2) = 1 \times 9 + 2 \times 4 + 2 \times 7 = 31$.

Bảng 2: Bảng tiện ích bên ngoài

Item	External Utility
AA	9
BB	11
CC	4
DD	6
EE	7

Định nghĩa 7 Tổng tiện ích của cơ sở dữ liệu D, ký hiệu là TU, như sau:

$$TU = \sum_{T_n \in D} tu(T_n) \tag{5}$$

Ví dụ trong Bảng 1, $TU = 41 + 31 + 149 + 121 + 85 + 83 + 137 + 134 = 781$.

Định nghĩa 8 Tầm quan trọng của itemsets được đánh giá bằng ngưỡng tiện ích tối thiểu. Giá trị của ngưỡng tiện ích tối thiểu, ký hiệu là $\delta = c(0 \leq c \leq TU)$, là một hằng số được chỉ định bởi chuyên gia.

Định nghĩa 9 Itemset X được coi là itemset tiện ích cao khi và chỉ khi tiện ích của itemset X không nhỏ hơn δ, ký hiệu là $HUI = \{X \subseteq I, u(X) \geq \delta\}$.

3.2 Tuyên bố vấn đề

Cho một cơ sở dữ liệu D, nó chứa các tập mục có tính tiện ích cao, được ký hiệu là HUI. Cho $S = \{S_1, S_2, \ldots, S_n\}$ là tập các tập mục hữu ích cao nhạy cảm và $NS(= HUIs - S)$ là tập các tập mục hữu ích cao không nhạy cảm. Mục đích của PPUM là ẩn tất cả các tập mục hữu ích cao nhạy cảm S. Cách chính để giải quyết vấn đề này là giảm giá trị tiện ích của $S_k (1 \leq k \leq n)$ và tạo ra các tập mục hữu ích cao không nhạy cảm (NS) dễ khám phá nhất có thể. Cơ sở dữ liệu sau khi dọn dẹp được ký hiệu là D'. $HUIs$ được khai thác từ D', ký hiệu là H'. Nói chung, quy trình của thuật toán PPUM như sau: 1) Tất cả các $HUIs$ được tìm thấy bằng thuật toán khai thác mẫu tiện ích; 2) Xác định tất cả các tập mục có tính tiện ích cao nhạy cảm; 3) Tất cả các tập mục hữu ích cao nhạy cảm đều bị ẩn bằng cách sử dụng thuật toán bảo vệ quyền riêng tư và giảm thiểu tác dụng phụ của thuật toán. Hiệu suất của PPUM được đo như sau.

Định nghĩa 10 Lỗi ẩn (HF) biểu thị các itemset có tính tiện ích cao nhạy cảm S vẫn được tìm thấy trong D' sau khi làm sạch. HF được tính bằng cách sử dụng phương trình (6).

$$HF = \frac{|S \cap H'|}{|H|} \tag{6}$$

Định nghĩa 11 Chi phí thiếu (MC) biểu thị các itemset có tính tiện ích cao không nhạy cảm bị thiếu, được định nghĩa là:

$$MC = \frac{|NS - H'|}{|NS|} \tag{7}$$

Định nghĩa 12 Chi phí nhân tạo (AC) thể hiện rằng các itemset trở thành $HUIs$ sau khi dọn dẹp, không phải là $HUIs$ trước khi dọn dẹp. AC được định nghĩa là:

$$AC = \frac{|H' - H|}{|H|} \tag{8}$$

Định nghĩa 13 Chi phí ẩn HidingCost : Lấy cảm hứng từ các thuật toán tiến hóa [43], bài viết đề xuất một khái niệm mới về HidingCost với nhiều ngưỡng để đo lường tác dụng phụ, được định nghĩa như sau:

$$\text{HidingCost} = w_1 \times HF + w_2 \times MC + w_3 \times AC \tag{9}$$

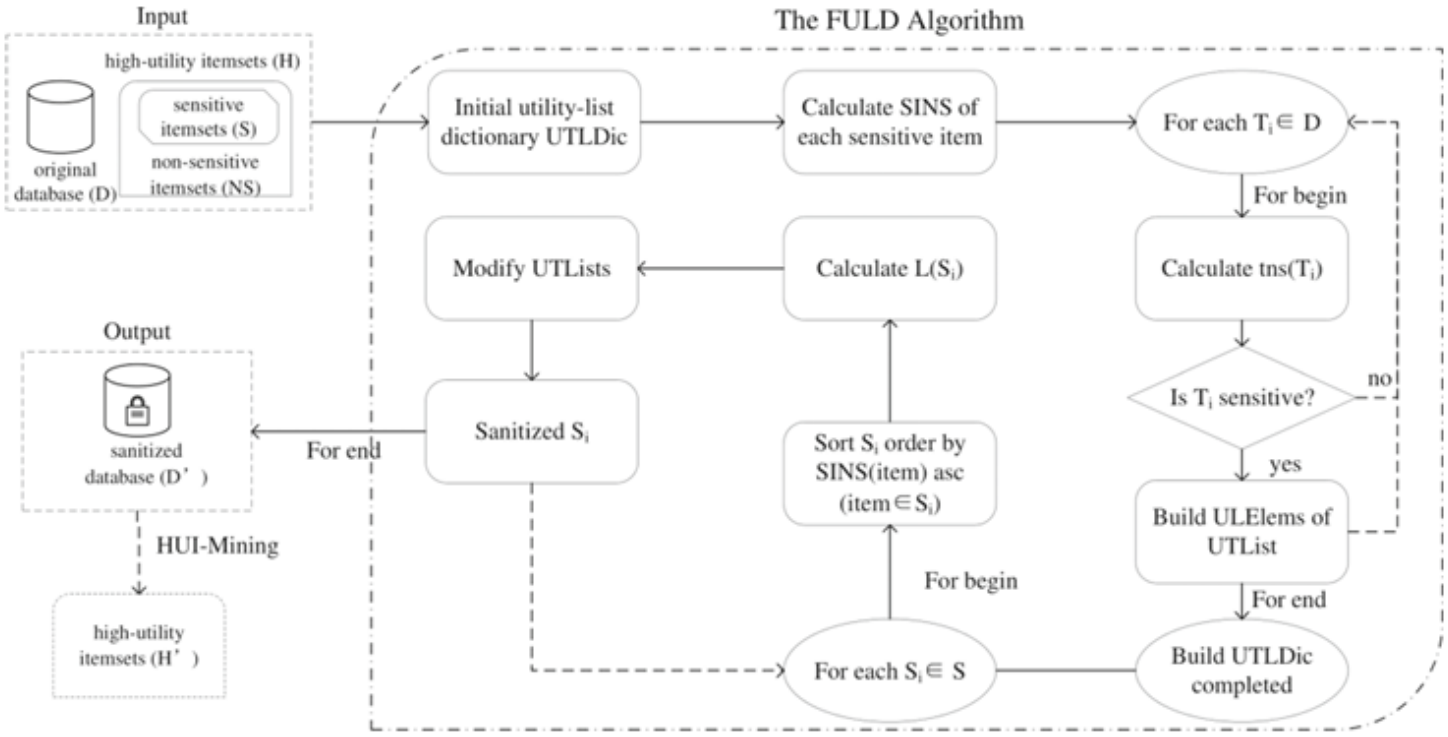
trong đó HF là lỗi ẩn, MC là chi phí thiếu, và AC là chi phí nhân tạo (xem phần 4.2). w_1, w_2, w_3 là trọng số tương đối của từng tác dụng phụ do người dùng xác định và $w_1 + w_2 + w_3 = 1$. Khi tỷ lệ ẩn của các tập mục nhạy cảm cao, w_1 là giá trị nhỏ hoặc thậm chí bằng 0. Người dùng chỉ định giá trị của w_2 và w_3 tương ứng theo tầm quan trọng tương đối của MC và AC.

4 Thuật toán đề xuất

Để ẩn các tập mục tiện ích nhạy cảm, nhiều thuật toán trước đó đề xuất các phương pháp khác nhau để giảm giá trị tiện ích. Mặc dù thuật toán FPUTT giảm số lần quét cơ sở dữ liệu bằng cách xây dựng cây nhưng việc xây dựng và bảo trì cây vẫn đòi hỏi nhiều chi phí. Trong phần này, cấu trúc từ điển danh sách tiện ích được đề xuất để ẩn thông tin nhạy cảm.Toàn bộ quá trình của thuật toán FULD được hiển thị trong Hình 1.

4.1 Từ điển danh sách tiện ích ban đầu

Bước đầu tiên của thuật toán FULD, bài báo xây dựng một từ điển danh sách tiện ích bao gồm các mục nhạy cảm và UTLList. Khi chủ sở hữu dữ liệu chia sẻ dữ liệu, bên thứ ba có thể tìm thấy thông tin quan trọng thông qua việc khai thác dữ liệu. Tuy nhiên, thông tin nhạy cảm có thể gây ra tổn thất nhất định cho người nắm giữ dữ liệu nếu được một bên thứ ba không đáng tin cậy lấy được. Vì vậy, bài báo đề xuất từ điển danh sách tiện ích (UTLDic) để lưu trữ những thông tin nhạy cảm có tính tiện ích cao. Để giảm thời gian quét cơ sở dữ liệu và số lượng tiện ích mục được tính toán trong quá trình dọn dẹp, thông tin về các mục nhạy cảm được lưu trữ trong UTLDic. Ngoài ra, bài báo còn



Hình 1: Quy trình tổng thể của FULD

Bảng 3: Bảng HUIs

Itemset	Utility	Itemset	Utility	Itemset	Utility	Itemset	Utility
ACDE	205	ACD	234	BE	202	BDE	250
BCDE	274	BCE	226	BD	289	BCD	325
BC	223	CDE	266	CE	305	CD	278

đề xuất các khái niệm SINS và tns để chọn mục nạn nhân, có thể làm giảm tác dụng phụ của thuật toán. So với các thuật toán khai thác tiện ích bảo vệ quyền riêng tư truyền thống, FULD có thể ẩn các tập mục nhạy cảm nhanh hơn và giảm tác dụng phụ. Các định nghĩa của FULD như sau:

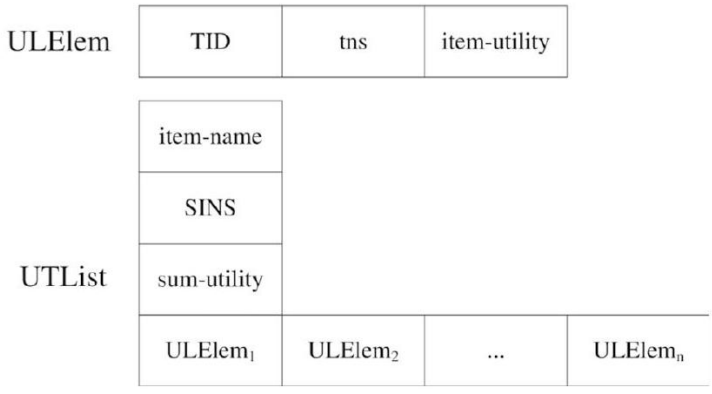
Định nghĩa 14 Số lần mỗi item nhạy cảm xuất hiện trong itemsets không nhạy cảm được ký hiệu là SINS(i_v). Ví dụ trong Bảng 3, giả định rằng {ACD} và {BC} là các itemsets có tính tiện ích cao nhạy cảm. Tập các item nhạy cảm là $\{A,B,C,D\}$. Theo Bảng 3, chỉ có một itemsets tiện ích cao không nhạy cảm {ACDE} chứa item nhạy cảm A. Do đó, $\text{SINS}(A) = 1$. Tương tự, $\text{SINS}(B) = 6$.

Định nghĩa 15 Mối quan hệ giữa transaction T_n và itemsets không nhạy cảm, ký hiệu là $\text{tns}(T_n)$, được trình bày dưới đây:

$$\text{tns}(T_n) = \frac{1}{1 + NSI\_num} \tag{10}$$

trong đó NSI_num biểu thị số lượng itemsets không nhạy cảm có trong T_n. Khi một giao dịch chứa nhiều item không nhạy cảm hơn thì tns sẽ nhỏ hơn. Ngược lại, tns lớn hơn. Lấy Bảng 1 làm ví dụ, T_5 chứa một itemset tiện ích cao không nhạy cảm $\{CD\}$. Do đó, $\text{tns}(T_5) = 1 \,/\, (1+1) = 0.5$.

Định nghĩa 16 Utility List Elem (ULElem) ULElem là một thành phần item nhạy cảm. Nó chứa ba trường: TID, tns (xem định nghĩa 15) và item-utility, trong đó TID là mã định danh duy nhất của transaction và item-utility là tiện ích của một item nhạy cảm. ULElem được hiển thị trong Hình 2.



Hình 2: ULElem và UTlist

Định nghĩa 17 Utility list (UTList) UTList chứa tất cả thông tin của item nhạy cảm S_i ($S_i \in S$) trong cơ sở dữ liệu D. Nó bao gồm bốn phần: item-name, SINS (xem Định nghĩa 14), sum-utility, và ULElems, trong đó item-name là tên item, và sum-utility là tổng tiện ích của item nhạy cảm trong cơ sở dữ liệu D. UTList được hiển thị trong Hình 2.

Định nghĩa 18 Utility-list dictionary structure (UTLDic) Utility Lists tạo nên cấu trúc UTLDic. Các khóa của UTLDic là một tập hợp các item nhạy cảm. UTLDic được hiển thị trong Hình 3.

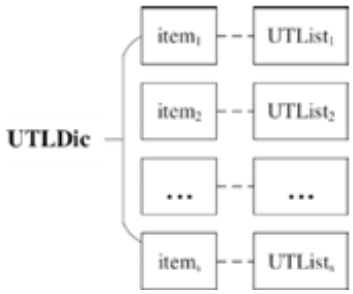
Định nghĩa 19 Giả sử $\cap$ là các ULElem chung giữa hai UTLList, trong đó các ULElem có cùng TID. Cho $L(X)$ là một tập hợp, được biểu diễn như sau:

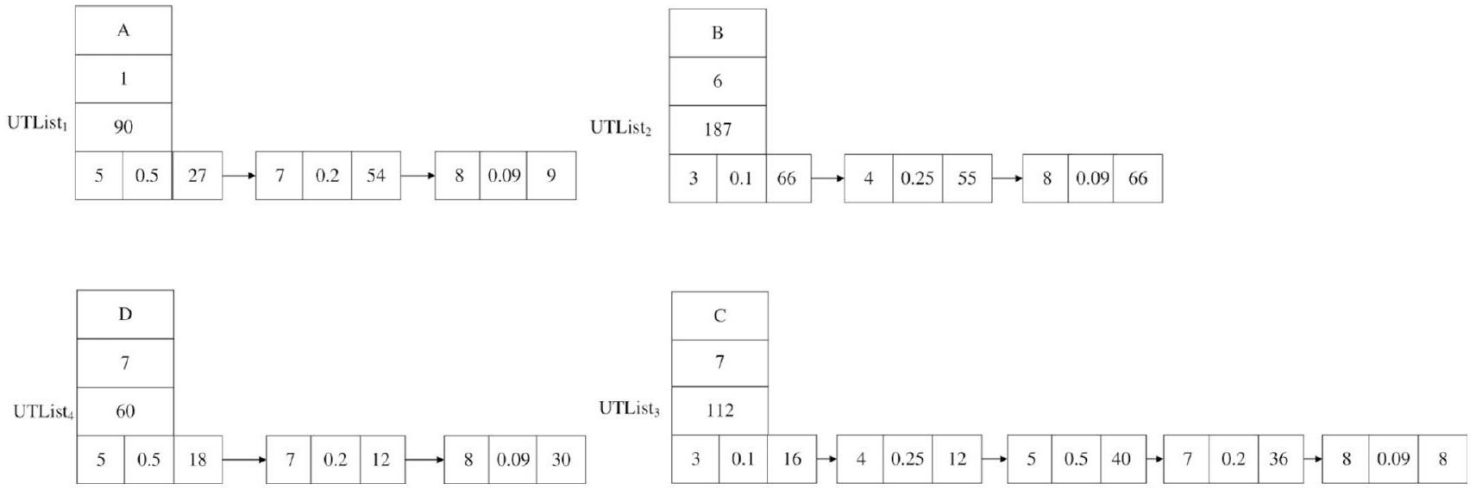
$$L = UTList_1 \cap UTList_2 \cap \cdots \cap UTList_n \tag{11}$$

Ví dụ trong Hình 4, $L(BC) = UTList_2 \cap UTList_3 = \{3,4\}$. $L(ACD) = UTList_1 \cap UTList_3 \cap UTList_4 = \{5,7,8\}$.

Theo Định nghĩa 16–18, bài báo đề xuất utility-list dictionary. Ví dụ trong Bảng 1, cấu trúc của UTLDic được thể hiện trong Hình 4 và 5. Ngoài ra, UTLDic được xây dựng và vệ sinh nhanh hơn FPUTT-tree. Trong [20], tác giả đã sử Header Table để trỏ tới nút của FPUTT-tree. Trong quá trình làm nhiễu FPUTT-tree, cần phải duyệt cây nhiều lần để tìm ra item tiện ích cao nhạy cảm nhất. Để giải quyết vấn đề này, bài báo đề xuất utility-list dictionary structure. Trong quá trình vệ sinh, không cần phải đi qua cây nhiều lần nữa. Thuật toán FULD chỉ cần tra cứu UTLDic để tìm các item nhạy cảm trong cơ sở dữ liệu D. UTList bao gồm item-name, SINS, sum-utility, và ULElems. Sau đó UTLList và các item nhạy cảm tạo nên UTLDic. Cho $\{ACD\}$ và $\{BC\}$ là các itemset nhạy cảm. Tập các item nhạy cảm là $\{A,B,C,D\}$, tạo nên các khóa của

Hình 3: The utility-list Dictionary





Hình 4: Một ví dụ về UTlists

UTLic. Lấy Bảng 1 làm ví dụ, cấu trúc của UTLDic được hiển thị trong Hình 4 và 5. Ví dụ, item nhạy cảm A trong Hình 5 trỏ đến $UTList_1$ trong Hình 4. Theo định nghĩa 14, $\text{SINS}(A) = 1$. Vì item nhạy cảm A của $\{ACD\}$ nằm trong T_5 , T_7 và T_8, các ULElem của $UTList_1$ là $(5,0,5,27)$, $(7,0,2,54)$ và $(8,0.09,9)$. Trong đó, $\text{tns}(ULElem.TID)$ được tính theo định nghĩa 15. $sum\_utility = 27 + 54 + 9 = 90$. Chi tiết về xây dựng utility-list dictionary được trình bày trong Thuật toán 1.

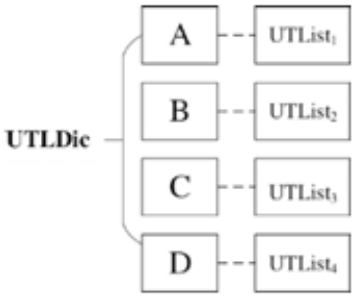
Trong Thuật toán 1, trước tiên, sự kết hợp của tất cả các itemset tiện ích cao nhạy cảm S được lấy và lưu trữ trong $SItem$ (dòng 1–4). Sau đó các item của $SItem$ sẽ được loại bỏ trùng lặp (dòng 5). Thứ hai, duyệt qua các item nhạy cảm $SItem$ và khởi tạo thông tin UTLDic (dòng 6–13). Sau đó tính $\text{SINS}(item)$ cho mỗi khóa của UTLDic (dòng 11). Thứ ba, cơ sở dữ liệu D được quét (dòng 14). Theo phương trình (10), $\text{tns}(T_i)$ được tính toán. Tập các item nhạy cảm SI được lấy từ T_i. Nó tạo thành các itemset tiện ích cao nhạy cảm (dòng 16). Cuối cùng, đi qua SI

Thuật toán 1: Xây dựng thuật toán UTLDic

Require: the database D, the sensitive high-utiltiy itemsets S, the non-sensitive high-utility itemsets NS
Ensure: the utility-list dictionary $UTLDic$

1: $SItem$={}
2: **for** each $S_i \in S$ **do**
3: $SItem$=$SItem \cup S_i$
4: **end for**
5: $SItem$= set($SItem$)
6: $UTLDic$=ϕ
7: **for** each $item \in SItem$ **do**
8: create a new node $UTList$
9: $UTList.item\_name$=$item$, $UTList.sum\_utility$=0
10: $UTList.ULElems$=ϕ, $UTList.SINS$=0
11: calculate $SINS(item)$ according to Definition 14
12: $UTLDic[item]$=$UTList$
13: **end for**
14: **for** each $T_i \in DB$ **do**
15: calculate $tns(T_i)$ using Eq. c
16: get sensitive items SI of T_i
17: **for** each $item \in SI$ **do**
18: create a new ULElem node ULE
19: $ULE.TID$=the TID of T_i, $ULE.utility$=$q(item, T_i) \times p(item)$
20: $ULE.tns = tns(T_i)$
21: $UTLDic[item].ULElems.append(ULE)$
22: $UTLDic[item].sum\_utility$ += $ULE.utility$
23: **end for**
24: **end for**
25: **return** $UTLDic$

Hình 5: Một ví dụ về UTLDic



và xây dựng UTList của UTLDic (dòng 17–23). ULElem được tạo theo TID, $u(item, T_i)$ và tns(T_i) (dòng 18-20). ULElem được thêm vào UTList và *sum_utility* của UTList được cập nhật (dòng 21–22). Toàn bộ quá trình xây dựng UTLDic đã kết thúc. Lấy Bảng 1 làm ví dụ, kết quả xây dựng UTLDic được thể hiện trong Hình 4 và 5.

4.2 Ẩn các itemset tiện ích cao nhạy cảm

Tiểu mục này mô tả chi tiết việc ẩn các itemset tiện ích cao nhạy cảm bằng UTLDic. Nguyên tắc vệ sinh là giữ tiện ích ở mức dưới δ bằng cách giảm tiện ích bên trong của item nhạy cảm. Các itemset nhạy cảm được sắp xếp theo thứ tự giảm dần của tiện ích. Các ULElem nạn nhân được sửa đổi theo các khái niệm về SINS và tns được đề xuất trong bài báo.

Trong quá trình dọn dẹp, UTLDic có thể được truy cập để tìm ULElem nạn nhân. Trong các công việc trước, cơ sở dữ liệu được quét nhiều lần. Để giải quyết vấn đề này, [20] đã đề xuất FPUTT-tree để giảm việc quét cơ sở dữ liệu nhiều lần trong quá trình nhiễu loạn. Dù sao đi nữa, vẫn có một số sai sót trong FPUTT-tree. FPUTT-tree được truy cập nhiều lần để tìm item tối đa. Nó lãng phí nhiều thời gian và gây ra tác dụng phụ cao. Ngoài ra, tiện ích của item được tính toán nhiều lần ở công việc trước. Thuật toán FULD được đề xuất để giải quyết các vấn đề này. Trong quá trình khử trùng, thuật toán không cần truy cập nhiều lần vào tất cả các item nhạy cảm. Ngoài ra, tiện ích của item không cần phải tính toán lại. Các khái niệm mới về SINS và tns được đề xuất trong bài báo làm giảm tác dụng phụ của FULD.

Trong Thuật toán 2, các itemset nhạy cảm được sắp xếp theo thứ tự giảm dần của tiện ích (dòng 1). Đầu tiên, đi ngang S (dòng 2). Theo SINS(item), các mục nhạy cảm của S_i được sắp xếp theo thứ tự tăng dần (dòng 3). Các TID của các transaction chứa S_i được lấy theo định nghĩa 19 (dòng 4). Thứ hai, tính toán tiện ích mục tiêu cần giảm (dòng 5). Các ULElem nạn nhân được sửa đổi cho đến khi targetUtil không lớn hơn 0 (dòng 6–22). Trong quá trình sửa đổi ULElem nạn nhân, các item nhạy cảm của S sẽ được truy cập (dòng 7). Khi truy cập một item nhạy cảm, các ULElem của UTlist được sắp xếp theo thứ tự giảm dần theo tns. Khi các giá trị tns giống nhau, chúng được sắp xếp tăng dần

Thuật toán 2: Thuật toán ẩn các itemset tiện ích cao nhạy cảm

Require: $UTLDic$, min_util, the sensitive high-utility itemsets S, the database D
Ensure: the sanitized $UTLDic$

1: sort S in descending order of $u(S_i)$ $(S_i \in S)$
2: **for** each $S_i \in S$ **do**
3: sort S_i in ascending order of $SINS(item)$ $(item \in S_i)$
4: calculate $l = L(S_i)$ according to Definition 19
5: $targetUtil$=$u(S_i)$-$min\_util + 1$
6: **while** $targetUtil > 0$ **do**
7: **for** each $item \in S_i$ **do**
8: $UTlist = UTLDic[item].UTList$
9: sort $ULElems$ of $UTlist$ order by tns desc, $utility$ asc
10: **for** each $elem \in UTlist$ and $elem \in l$ and $targetUtil > 0$ **do**
11: **if** $elem.utility \leq targetUtil$ **then**
12: $targetUtil- = elem.utility$
13: $elem.utility$=0
14: **else**
15: $count = q(item, elem.TID) - \lceil \frac{targetUtil}{p(item)} \rceil$
16: $elem.utility = count \times p(item)$
17: $targetUtil = 0$
18: **end if**
19: update $UTLDic[item].sum\_utility$
20: **end for**
21: **end for**
22: **end while**
23: **end for**

thứ tự của tiện ích (dòng 8–9). Sau đó lần lượt truy cập ULElem của UTlist (dòng 10). Nếu ULElem chứa trong các transaction có TID nằm trong $L(S_i)$, thì ULElem được sửa đổi theo công thức (dòng 11–18). Cuối cùng, UTLDic được cập nhật (dòng 19).

Ví dụ: Trong Bảng 3, chúng tôi giả sử rằng $\{ACD\}$ và $\{BC\}$ là các itemset tiện ích cao nhạy cảm, trong đó $u(ACD)$ là 234 và $u(BC)$ là 223. Các itemset nhạy cảm này được sắp xếp theo thứ tự giảm dần của tiện ích như sau: $\{ACD\}$ và $\{BC\}$. $\{ACD\}$ được truy cập lần đầu tiên. Khi đó $\{ACD\}$ được sắp xếp theo thứ tự tăng dần của SINS(item) như sau: $\{ADC\}$. Trong bài báo, ngưỡng tiện ích tối thiểu δ được đặt thành 200. Theo định nghĩa 19, TID của các transaction chứa $\{ACD\}$ là $L(ACD) = \{5,7,8\}$. Tổng tiện ích cần sửa đổi là $u(ACD,T_5) + u(ACD,T_7) + u(ACD,T_8) - \delta = 234 - 200 + 1 = 35$. Thứ hai, A được thăm. Theo Hình 5, có $UTList_1$. Các ULElem của $UTList_1$ được sắp xếp theo thứ tự giảm dần theo tns. Khi các giá trị tns giống nhau, chúng được sắp xếp theo thứ tự tiện ích tăng dần. Sau đó theo thứ tự, các ULElem lần lượt được truy cập. Vì $27 < 35$ nên tiện ích của ULElem $(5,0.5,27)$ được đổi thành 0 và $sum\_utility$ của $UTList_1$ được sửa thành $90 - 27 = 63$. targetUtil bằng $35 - 27 = 8$. Tiếp theo, ULElem $(7,0.2,54)$ được truy cập. Vì $8 < 54$ nên tiện ích của ULElem này được cập nhật thành $(6 - \lceil 8/9 \rceil) \times 9 = 45$. Giá trị của targetUtil là 0. Hoạt động của $\{BC\}$ tương tự như Hình 6 ở trên. Kết quả vệ sinh cuối cùng được thể hiện trong Hình 7.

4.3 Thuật toán FULD

Tiểu mục này giới thiệu toàn bộ quá trình của thuật toán FULD. So với các công việc liên quan trước đây, thuật toán không cần phải truy cập và quét cơ sở dữ liệu nhiều lần khi làm xáo trộn cơ sở dữ liệu gốc. Như đã mô tả ở thuật toán 2, thuật toán chỉ cần truy cập vào utility-list dictionary để sửa đổi tiện ích của item. Trong thuật toán FULD, cơ sở dữ liệu gốc được truy cập để xây dựng UTLDic. Sau đó duyệt qua các item nhạy cảm của itemset nhạy cảm để dọn dẹp theo Mục 4.2. Cuối cùng, thuật toán tạo ra cơ sở dữ liệu DB' đã được làm sạch. Thuật toán FULD được thể hiện trong Thuật toán 3.

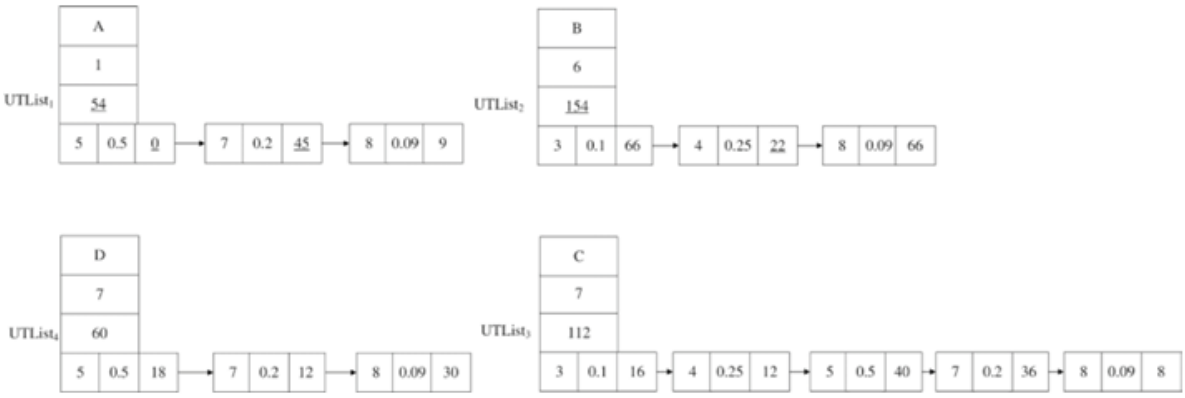
5 Thí nghiệm

Các thí nghiệm chứa bốn bộ dữ liệu. Đặc điểm của các bộ dữ liệu được thể hiện trong Bảng 4. Những bộ dữ liệu này có thể được lấy từ SPMF (http://www.philippe-fournier-viger.com/spmf/). Density biểu thị độ thưa thớt của tập dữ liệu. AvgLen biểu thị số lượng các transaction. Công thức tính toán như sau:

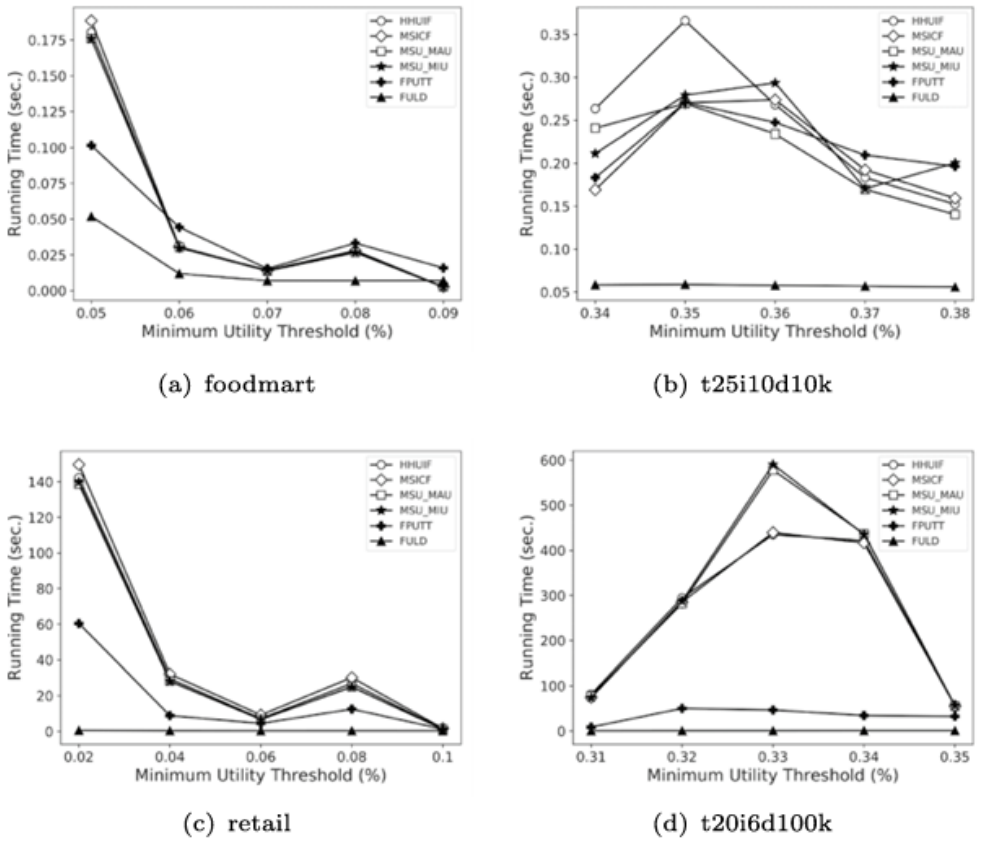
$$\text{Density} = \frac{\text{AvgLen}}{\text{Itemcount}} \tag{12}$$

Bên cạnh việc thử nghiệm trên các bộ dữ liệu khác nhau, bài báo còn so sánh các thuật toán trước đó như: HHUIF và MSICF [18], MSU-MAU và MSU-MIU [19], FPUTT [20]. Trong thử nghiệm sau, chúng tôi đã sử dụng ngưỡng δ khác nhau và tỷ lệ phần trăm thông tin nhạy cảm khác nhau cho các thử nghiệm so sánh. Các itemset nhạy cảm trong thử nghiệm này được chọn ngẫu nhiên từ HUI. Thuật toán được sử dụng để khai thác các mẫu tiện ích cao là ULB-Miner [44]. Do các bộ dữ liệu thử nghiệm không chứa tiện ích bên trong và bên ngoài nên chúng tôi đã áp dụng phân phối uniform được đề xuất bởi Ge et al.[45] để gán giá trị tiện ích nội bộ cho từng mục trong transaction. Hơn nữa, chúng tôi đã sử dụng phân phối Gaussian được đề xuất bởi Tassa et al.[46] để tạo ra tiện ích bên ngoài tương ứng với từng item duy nhất. Các thử nghiệm được thực hiện trên Windows 10 với CPU Intel Core i7 và RAM 16 GB.

Bên cạnh việc thử nghiệm trên các bộ dữ liệu khác nhau, bài viết này còn so sánh các thuật toán trước đó như HHUIF và MSICF, MSU-MAU và MSU-MIU và FPUTT. Trong



Hình 6: Một ví dụ về UTLDic đã được làm sạch



Hình 7: So sánh thời gian chạy theo tỷ lệ phần trăm thông tin nhạy cảm khác nhau

các thử nghiệm sau, chúng tôi đã sử dụng ngưỡng δ khác nhau và tỷ lệ phần trăm thông tin nhạy cảm khác nhau cho các thử nghiệm so sánh. Các itemset nhạy cảm trong thử nghiệm này được chọn ngẫu nhiên từ HUI. Thuật toán được sử dụng để khai thác các mẫu tiện ích cao là ULB-Miner [44]. Do các bộ dữ liệu thử nghiệm không chứa tiện ích bên trong và bên ngoài nên chúng tôi đã áp dụng phân phối uniform được đề xuất bởi Ge et al.[45] để gán giá trị tiện ích nội bộ cho từng mục trong transaction. Hơn nữa, chúng tôi đã sử dụng phân phối Gaussian được đề xuất bởi Tassa et al.[46] để tạo ra tiện ích bên ngoài tương ứng với từng item duy nhất. Các thử nghiệm được thực hiện trên Windows 10 với CPU Intel Core i7 và RAM 16 GB.

5.1 Phân tích thời gian chạy

Tiểu mục này giới thiệu so sánh hiệu quả hoạt động của sáu thuật toán dưới các bộ dữ liệu khác nhau, ngưỡng tiện ích tối thiểu khác nhau và tỷ lệ phần trăm thông tin nhạy cảm khác nhau, như được hiển thị trong Hình 7 và 8.

Trong khai thác mẫu có tính tiện ích cao, các itemset tiện ích cao nhạy cảm chỉ là một phần nhỏ của các itemset tiện ích cao. FULD chỉ lưu trữ các ULElem, nơi chứa các item nhạy cảm, không phải tất cả các transaction có chứa các item nhạy cảm. FULD nén và lưu trữ thông tin nhạy cảm có liên quan vào ULDic. ULDic được đề xuất trong bài báo chủ yếu được lưu trữ trong bộ nhớ và không gian dành cho thông tin nhạy cảm được nén là nhỏ.

Như được hiển thị trong Hình 7, thời gian chạy của thuật toán cho thấy xu hướng giảm khi ngưỡng tiện ích tối thiểu tăng lên. Lý do là khi ngưỡng tiện ích tối thiểu tăng lên thì số lượng HUIs sẽ giảm đi. Sự cố này dẫn đến tỷ lệ phần trăm HUIs giảm và thời gian chạy của thuật toán giảm. Vì các itemset nhạy cảm được chọn ngẫu nhiên, nên số lượng

Thuật toán 3: Thuật toán FULD

Require: the original database DB, the sensitive high-utility itemsets S, δ
Ensure: the sanitized database DB'

1: call Construct UTLDic Algorithm
2: call Hide Sensitive High-utility Itemsets Algorithm
3: generate the sanitized database DB'

Bảng 4: Bộ dữ liệu thực nghiệm

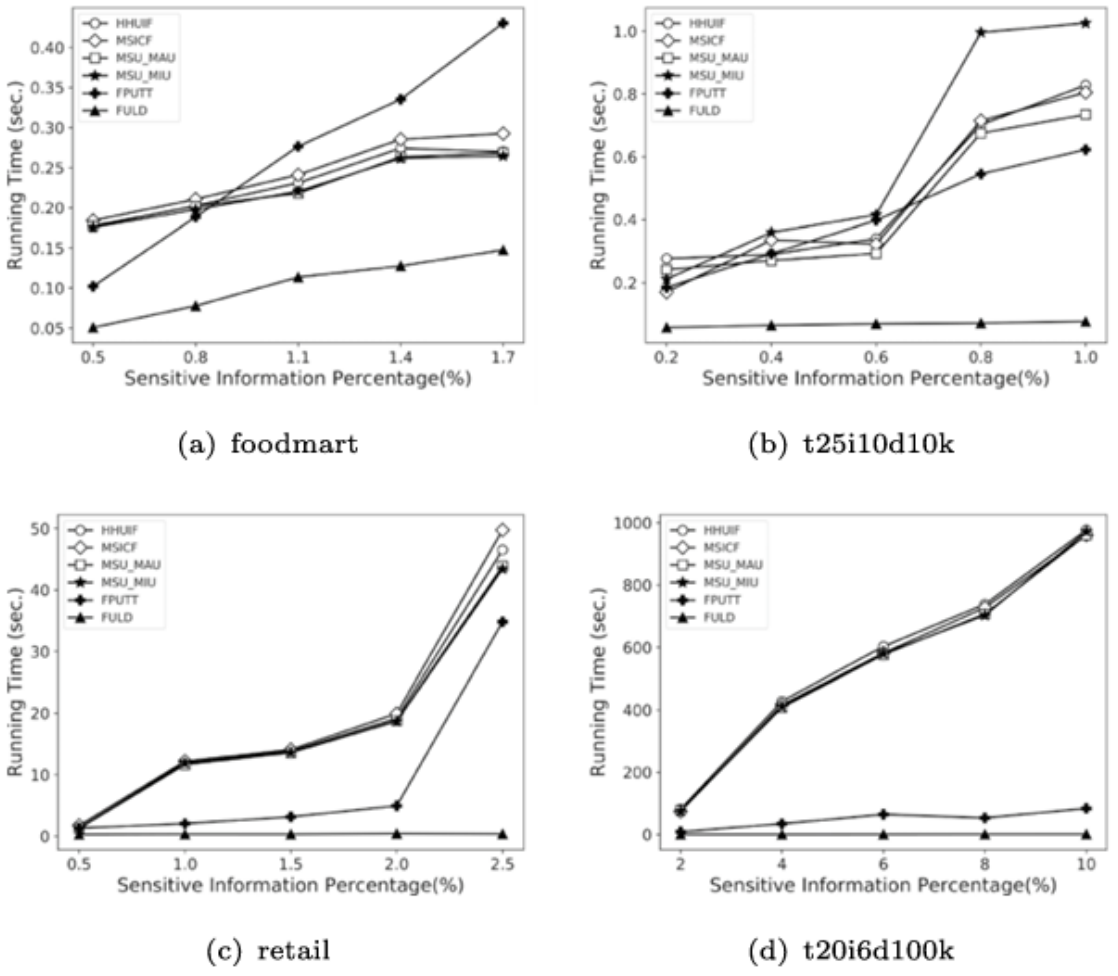
Dataset	|Dataset|	|Item|	AvgLen	Density
foodmart	4141	1559	4.42	0.28%
t25i10d10k	9976	929	24.77	2.67%
retail	88,162	16,470	1030.00	6.25%
t20i6d100k	99,922	893	24.77	2.77%

của các transaction có itemset tiện ích cao nhạy cảm có thể khác nhau. Càng nhiều transaction chứa các itemset nhạy cảm thì quá trình dọn dẹp càng mất nhiều thời gian. Ví dụ, tập dữ liệu t20i6d100k trong Hình 7 vẫn không ảnh hưởng đến xu hướng giảm chung của thời gian chạy thuật toán khi ngưỡng tiện ích tối thiểu tăng lên. Khi các tập dữ liệu và ngưỡng tiện ích tối thiểu giống nhau, thuật toán FULD được đề xuất trong bài viết này tốt hơn đáng kể so với năm thuật toán còn lại. Thời gian chạy của FULD ngắn hơn FPUTT từ 15–20 lần. Trong PPUM, chúng ta nên giảm thiểu các tác dụng phụ và cân nhắc thời gian chạy. Khi tập dữ liệu lớn, thời gian chạy sẽ lâu hơn hàng chục, thậm chí hàng trăm lần so với tập dữ liệu nhỏ hơn. Nếu nó tiêu tốn nhiều thời gian vì ít tác dụng phụ thì cũng không được khuyến khích cho các nhà nghiên cứu. Để giải quyết vấn đề, chúng tôi đề xuất thuật toán FULD. Trong Hình 7, khi tỷ lệ phần trăm thông tin nhạy cảm tăng lên thì thời gian chạy của các thuật toán khác nhau cũng tăng lên. Bởi vì số lượng SHUIs càng lớn thì càng mất nhiều thời gian để làm xáo trộn cơ sở dữ liệu. Khi tập dữ liệu và tỷ lệ phần trăm thông tin nhạy cảm bằng nhau thì hiệu suất của thuật toán FULD sẽ cao hơn.

Cho dù đó là Hình 7 hay Hình 8, số lượng giao dịch trong tập dữ liệu càng thấp thì thuật toán chạy càng ít, chẳng hạn như bộ dữ liệu foodmart và t25i10d10k. So với hai bộ dữ liệu còn lại, chúng chỉ có vài nghìn giao dịch nên thời gian chạy tương đối ngắn. Theo mật độ tập dữ liệu trong Bảng 4, các thử nghiệm cho thấy dù tập dữ liệu dày đặc hay thưa thớt, thuật toán FULD có khả năng mở rộng tốt và hiệu quả vận hành tốt hơn năm thuật toán còn lại.

5.2 Tác dụng phụ

Trong tiểu mục này, chúng tôi giới thiệu kết quả thử nghiệm của tất cả các thuật toán về mặt tác dụng phụ. Các tác dụng phụ được thảo luận trong bài viết này được đo lường bằng Hiding Cost. Nó bao gồm hiding failure, missing cost, và artificial cost (xem mục 3.2).



Hình 8: So sánh thời gian chạy dưới các ngưỡng tiện ích tối thiểu khác nhau

HidingCost áp dụng cho mọi tình huống. Khi thuật toán không dẫn đến hiding failure, w_1 có thể được đặt thành 0 hoặc giá trị nhỏ hơn. Các giá trị của w_2 và w_3 được đặt tương tự. Trong các thử nghiệm, w_2 và w_3 được đặt thành 0.4 và 0.1 cho tất cả các thuật toán. Vì việc hiding failure quan trọng hơn nên w_1 được đặt thành 0.5.

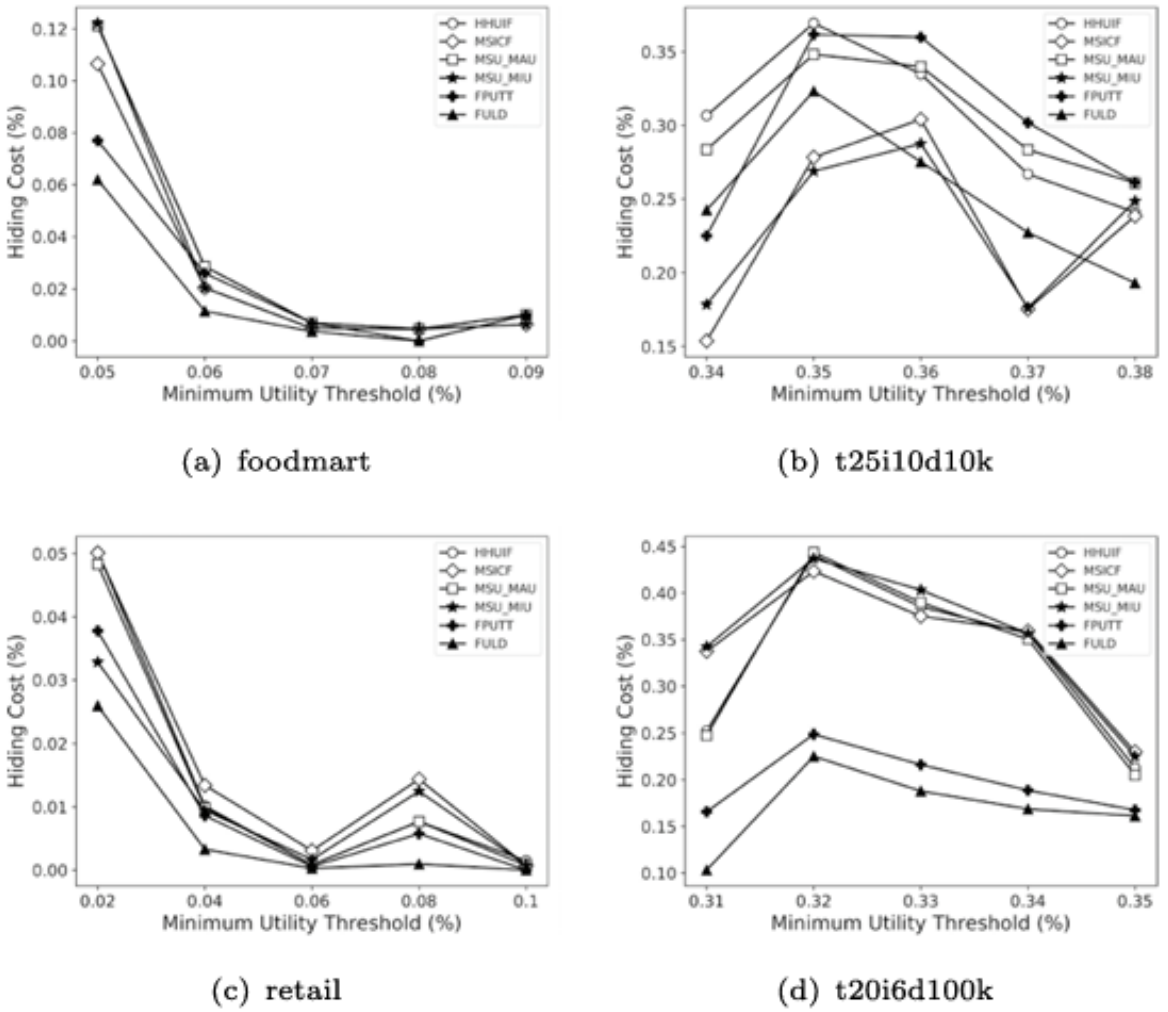
Trong quá trình dọn dẹp, FULD ưu tiên chọn item nhạy cảm có SINS(item) nhỏ hơn để sửa đổi. SINS(item) càng nhỏ thì item đó càng ít xuất hiện trong các itemset tiện ích cao không nhạy cảm. Điều này có thể làm giảm tác động lên các itemset tiện ích cao (NS) không nhạy cảm. Khi ULElem nạn nhân được chọn, ULElem có $\text{tns}(T_i)$ lớn hơn sẽ được ưu tiên. Bởi vì $\text{tns}(T_i)$ càng lớn thì NS mà T_i chứa càng ít. Nói cách khác, khi hạng mục được sửa đổi thì tác động lên NS sẽ ít hơn. Để giảm tác động đến các HUIs khác, ULElem với ít tiện ích hơn sẽ được ưu tiên hơn.

Bài viết so sánh hiding cost của sáu thuật toán trong các điều kiện khác nhau.Hình 9 và 10 thể hiện kết quả thử nghiệm. Hình 9 cho thấy tác dụng phụ của FULD tương đối nhỏ so với các thuật toán khác ở các ngưỡng tiện ích tối thiểu khác nhau. Khi ngưỡng tiện ích tối thiểu tăng lên, tác dụng phụ của tất cả các thuật toán sẽ giảm đi. Trong Hình 10, FULD vẫn có hiệu quả tốt hơn về mặt tác dụng phụ. Khi tỷ lệ phần trăm thông tin nhạy cảm tăng lên, hiding costs tăng lên. Thông tin càng nhạy cảm cần được giấu đi thì tác dụng phụ càng cao. Tóm lại, thuật toán FULD không chỉ có thời gian chạy ngắn hơn mà còn tạo ra ít tác dụng phụ hơn.

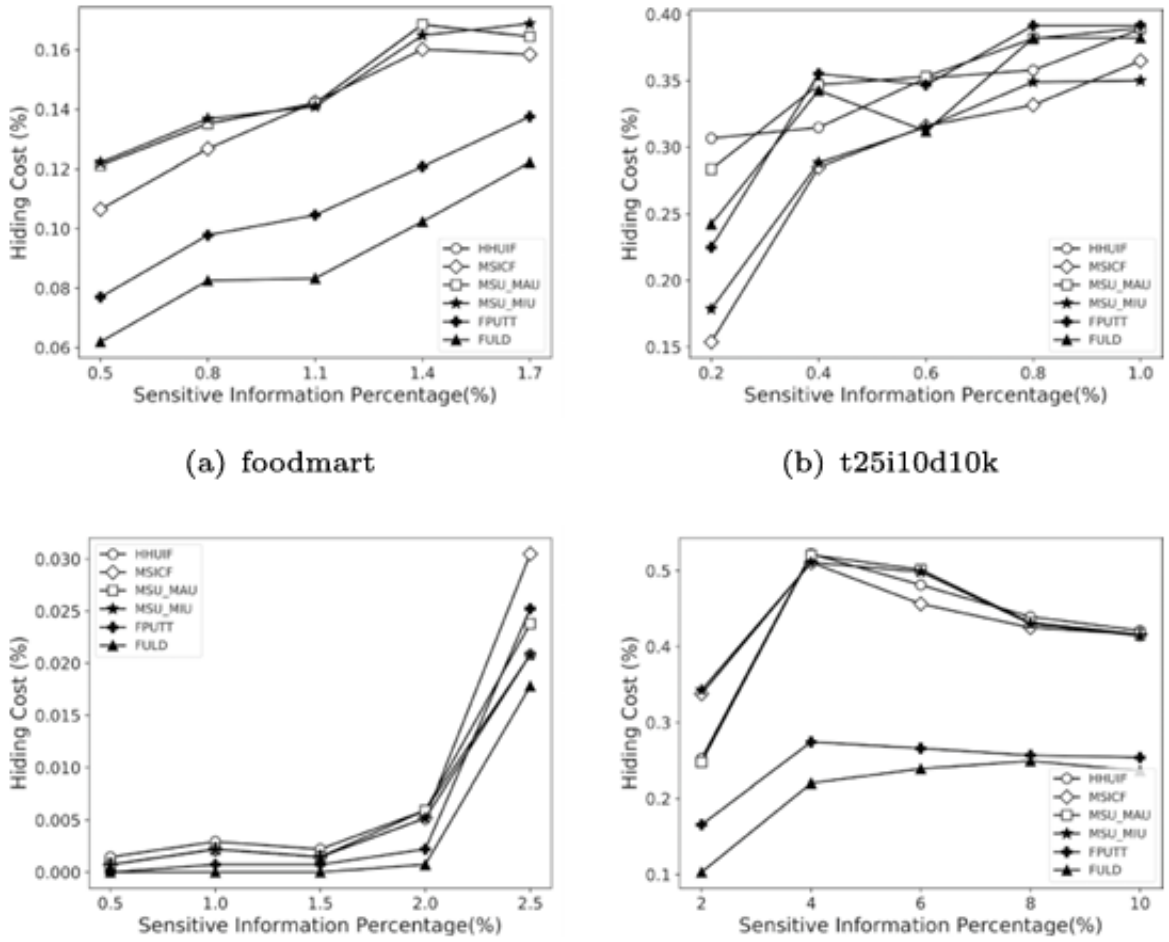
6 Phân tích độ phức tạp

Trong phần này, độ phức tạp về thời gian và không gian của thuật toán FPUTT và FULD được thực hiện để so sánh về mặt lý thuyết. Các thông số liên quan như sau:

- N_s là số lượng itemset tiện ích cao nhạy cảm;
- DB_{scan} là số lượng item cần truy cập khi quét cơ sở dữ liệu;
- $NR(S_p)$ là số lần lặp lại cần thiết để ẩn một itemset nhạy cảm S_p;
- $TC(k, S_p)$ là thời gian cần thiết để cập nhật tiện ích của S_p trong lần quét cơ sở dữ liệu thứ k;
- $TREE_{\text{scan}}$ là số nút được truy cập trong một lần duyệt cây.
- $UTLIST_{\text{scan}}$ là số lượng ULElem được truy cập trong một lần truyền tải UTList.



Hình 9: So sánh chi phí ẩn dưới các ngưỡng tiện ích tối thiểu khác nhau



Hình 10: So sánh Hiding Cost theo tỷ lệ phần trăm thông tin nhạy cảm khác nhau

Dựa trên phân tích mã giả của thuật toán FPUTT [20] và FULD, chúng tôi đã rút ra công thức độ phức tạp thời gian sau:

$$\text{FPUTT} \rightarrow 3 * DB_{\text{scan}} + \sum_{p=1}^{N_s} \sum_{k=1}^{NR(S_p)} \{TREE_{\text{scan}} + TC(k, S_p)\},$$

$$\text{FULD} \rightarrow 2 * DB_{\text{scan}} + \sum_{r=1}^{N_s} \sum_{k=1}^{NR(S_r)} \{UTLIST_{\text{scan}} + TC(k, S_r)\}.$$

Thuật toán FULD xây dựng UTLDic bằng một lần quét cơ sở dữ liệu (DB_{scan}) và truy cập UTLDic $\sum_{r=1}^{N_s} \sum_{k=1}^{NR(S_r)} \{UTLIST_{\text{scan}}\}$. Sau đó các item nhạy cảm trong UTLDic được cập nhật $\left(\sum_{r=1}^{N_S} \sum_{k=1}^{NR(S_r)} \{TC(k, S_r)\}\right)$. Tuy nhiên, thuật toán FPUTT cần 2 lần quét cơ sở dữ liệu ($2 * DB_{\text{scan}}$) để dựng cây và đi ngang qua cây $\left(\sum_{p=1}^{N_s} \sum_{k=1}^{NR(S_p)} \{TREE_{\text{scan}}\}\right)$. Lưu ý rằng $\exists NR(S_r) \neq NR(S_p)$ và $UTLIST_{\text{scan}}$ $TREE_{\text{scan}}$. Theo bài báo [20], $TREE_{\text{scan}}$ tệ nhất được tính như sau:

$$TREE_{\text{scan}} = N_n \times \frac{\sum_{j=1}^{N_n} LN(n_j)}{N_n} \times \frac{\sum_{p=1}^{N_s} LS(n_p)}{N_s T_s} \quad (13)$$

N_n là số nút trong cây. T_s là số lượng transaction với nhiều itemset nhạy cảm. Trong mỗi itemset nhạy cảm S_p của SI-table, độ dài của tập TID được ký hiệu là $LS(S_p)$. $LN(n_j)$ là độ dài của tập TID chứa trong mỗi nút của cây. N_s là số lượng các itemset nhạy cảm. Gọi N_e là số ULElem trong UT-List. $UTLIST_{\text{scan}}$ tệ nhất được tính như sau:

$$UTLIST_{\text{scan}} = N_e \quad (14)$$

Rõ ràng, $UTLIST_{\text{scan}}$ nhỏ hơn $TREE_{\text{scan}}$. Theo phân tích ở trên, độ phức tạp về thời gian của FPUTT là: $O(DB_{\text{scan}} + N_s \times NR(S_p) \times TREE_{\text{scan}})$. Độ phức tạp thời gian của FULD là: $O(DB_{scan} + N_s \times NR(S_r) \times UTLIST_{scan})$. Do đó, thuật toán FULD có hiệu suất tốt hơn trong thời gian chạy. Đặt N_{si} là độ dài của tập item nhạy cảm, và LV là độ dài tập TID của nút trong FPUTT-tree. Độ phức tạp về không gian của thuật toán FPUTT và FULD như sau: $FPUTT = O(N_{si} + N_n \times LN_{\text{avg}})$, và $FULD = O(N_{si} + N_{si} \times N_e)$.

7 Kết luận

Với sự nâng cao nhận thức về quyền riêng tư của mọi người, khả năng vi phạm quyền riêng tư do HUPM gây ra đã thu hút sự chú ý của các nhà nghiên cứu. Nói tóm lại, PPUM đang ẩn các itemset tiện ích cao nhạy cảm. Khi các công ty chia sẻ dữ liệu, các bên thứ ba không thể khai thác thông tin nhạy cảm ẩn. Trong bài báo, chúng tôi đề xuất

thuật toán FULD, cố gắng giảm thiểu tác dụng phụ trong khi vẫn đảm bảo quyền riêng tư. Chúng tôi sử dụng từ điển danh sách tiện ích để cải thiện quy trình vệ sinh. Thông qua phân tích và so sánh thử nghiệm sâu rộng, thuật toán của chúng tôi có hiệu suất vượt trội về thời gian chạy và tác dụng phụ. Trong các nghiên cứu trong tương lai, các nhà nghiên cứu nên chú ý đến việc làm giảm tác dụng phụ và thời gian hoạt động của PPUM. Một mặt, chiến lược dựa trên cây có thể cải thiện quá trình khử trùng. Đó là một hướng nghiên cứu quan trọng nhằm thiết kế một chiến lược cắt tỉa mới để giảm không gian tìm kiếm. Mặt khác, mặc dù thuật toán chính xác dựa trên quy hoạch tuyến tính số nguyên có thể tìm ra lời giải tối ưu nhưng phải mất một thời gian dài. Việc kết hợp thuật toán heuristic để tìm ra lời giải tối ưu có tính đến thời gian chạy cũng là một hướng nghiên cứu mới.

Acknowledgments Các tác giả xin chân thành cảm ơn người biên tập và những người phản biện ẩn danh vì những nhận xét sâu sắc và mang tính xây dựng của họ, đã giúp chúng tôi cải thiện công việc này rất nhiều.

Author Contributions Tất cả các tác giả đều đóng góp vào ý tưởng và thiết kế nghiên cứu. Việc chuẩn bị tài liệu, thu thập và phân tích dữ liệu được thực hiện bởi Chunyong Yin và Ying Li. Bản thảo đầu tiên của bản thảo được viết bởi Ying Li và tất cả các tác giả đều nhận xét về các phiên bản trước của bản thảo. Tất cả các tác giả đều đọc và phê duyệt bản thảo cuối cùng.

Funding Nghiên cứu này được tài trợ bởi Quỹ khoa học tự nhiên quốc gia Trung Quốc (61772282). Nó cũng được hỗ trợ bởi Chương trình đổi mới thực hành và nghiên cứu sau đại học của tỉnh Giang Tô (KYCX21 1008).

Declarations

Conflict of interest Các tác giả tuyên bố rằng họ không có lợi ích tài chính cạnh tranh hoặc mối quan hệ cá nhân nào có thể ảnh hưởng đến công việc được báo cáo trong bài viết này.

References