

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

TRẦN KHẮC BÌNH

BẢO VỆ TÍNH RIÊNG TỰ TRONG KHAI
THÁC MẪU DỰA TRÊN PHƯƠNG
PHÁP TỐI ƯU NGẪU NHIÊN

KHÓA LUẬN TỐT NGHIỆP CỦ NHÂN CNTT
CHƯƠNG TRÌNH CHÍNH QUY CHUẨN

Tp. Hồ Chí Minh, tháng 3/2025

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

TRẦN KHẮC BÌNH - 20120437

BẢO VỆ TÍNH RIÊNG TỰ TRONG KHAI
THÁC MẪU DỰA TRÊN PHƯƠNG
PHÁP TỐI ƯU NGÃU NHIÊN

KHÓA LUẬN TỐT NGHIỆP CỦ NHÂN CNTT
CHƯƠNG TRÌNH CHÍNH QUY CHUẨN

GIẢNG VIÊN HƯỚNG DẪN

ThS. Nguyễn Ngọc Đức

Tp. Hồ Chí Minh, tháng 3/2025

Lời cam đoan

Tôi xin cam đoan đây là công trình nghiên cứu của tôi dưới sự hướng dẫn của ThS. Nguyễn Ngọc Đức. Tất cả dữ liệu, phân tích và kết luận trong nghiên cứu đều được thực hiện một cách trung thực, khách quan, không sao chép từ bất kỳ nguồn nào mà không trích dẫn rõ ràng. Kết quả thực nghiệm trong nghiên cứu này chưa từng được công bố trong bất kỳ công trình nào trước đó. Mọi tài liệu tham khảo đều được tôi sử dụng đúng quy định, đảm bảo tính minh bạch và khoa học.

BẢN NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP (HƯỚNG NGHIÊN CỨU)

Tên đề tài : BẢO VỆ TÍNH RIÊNG TƯ TRONG KHAI THÁC MẪU DỰA TRÊN PHƯƠNG PHÁP TỐI ƯU NGẦU NGHIÊN.

Sinh viên thực hiện : Trần Khắc Bình – 20120437.

Giảng viên hướng dẫn: Nguyễn Ngọc Đức.

1. Chủ đề và ý tưởng nghiên cứu:

Luận văn tập trung vào một vấn đề đang được quan tâm trong lĩnh vực khai thác dữ liệu: bảo vệ tính riêng tư trong khai thác mẫu hữu ích (PPUM). Sinh viên đã xác định rõ động lực nghiên cứu từ nhu cầu cần bằng giữa việc khai thác thông tin hữu ích và bảo vệ thông tin nhạy cảm. Ý tưởng chính của luận văn là phát triển một thuật toán mới (SO2DI) kết hợp tối ưu hóa ngẫu nhiên với chiến lược xóa item để ẩn các mẫu hữu ích nhạy cảm, đồng thời giảm thiểu tác dụng phụ. Hướng tiếp cận này thể hiện tính sáng tạo và khả năng ứng dụng cao trong thực tiễn.

2. Phương pháp nghiên cứu:

Sinh viên nghiên cứu và đề xuất phương pháp ẩn dữ liệu mới SO2DI với ý tưởng sử dụng các phương pháp tối ưu ngẫu nhiên. Bên cạnh đó sinh viên còn thiết kế một hàm mục tiêu hiệu quả (f_{SO2DI}) giúp cân bằng giữa ẩn thông tin nhạy cảm và ảnh hưởng phụ.

3. Đóng góp Khoa học và thực tiễn:

Trong khóa luận, sinh viên đề xuất thuật toán ẩn dữ liệu SO2DI. Kết quả nghiên cứu có thể ứng dụng trong các hệ thống bảo vệ thông tin nhạy cảm và khai thác dữ liệu.

4. Quá trình thực hiện:

Sinh viên nỗ lực thực hiện đề tài và chủ động trao đổi với giáo viên trong suốt quá trình thực hiện khóa luận.

5. Báo cáo viết:

Báo cáo gồm 6 chương được trình bày rõ ràng. Nội dung đầy đủ và khách quan.

6. Trình bày trước hội đồng:

Sinh viên nắm rõ vấn đề khi trình bày trước hội đồng.

7. Công bố khoa học/ Ứng dụng thực tế:

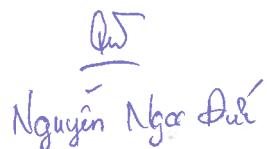
Khóa luận chưa có công bố khoa học.

Đánh giá xếp loại: Đạt yêu cầu.

TP.HCM, ngày 14 tháng 03 năm 2025

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)


Nguyễn Ngọc Diệp

BẢN NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP (HƯỚNG NGHIÊN CỨU)

Tên đề tài : Bảo vệ tính riêng tư trong khai thác mẫu dựa trên phương pháp tối ưu ngẫu nhiên

Sinh viên thực hiện : 20120437 – Trần Khắc Bình

Giảng viên phản biện: ThS. Phạm Trọng Nghĩa.....

1. Chủ đề và ý tưởng nghiên cứu:

Bài toán Khai thác mẫu hữu ích (KTMHU) là bài toán quan trọng khám phá tri thức từ dữ liệu. Tuy nhiên, quá trình khai thác này có thể vô tình tiết lộ thông tin nhạy cảm, gây ra những rủi ro về quyền riêng tư. Để giải quyết vấn đề này, lĩnh vực bảo vệ tính riêng tư trong khai thác mẫu hữu ích (PPUM) ra đời nhằm che giấu các mẫu hữu ích nhưng nhạy cảm trước khi công bố hoặc sử dụng dữ liệu.

Một thách thức lớn của KTMHU là sau quá trình bảo vệ, dữ liệu có thể bị biến dạng, làm giảm chất lượng khai thác và ảnh hưởng đến hiệu quả của HUPM

Khóa luận nghiên cứu và đề xuất một thuật KTMHU mới có tên là SO2DI, kết hợp tối ưu hóa ngẫu nhiên với chiến lược xóa item để làm nhiễu dữ liệu hiệu quả, đảm bảo che giấu các mẫu nhạy cảm mà vẫn duy trì tính hữu ích của dữ liệu.

2. Phương pháp nghiên cứu:

Trên cơ sở khảo sát các thuật toán bảo vệ tính riêng tư trong KTMHU hiện tại, tác giả nhận thấy các thuật toán này vẫn gặp hạn chế về hiệu suất xử lý và có thể gây ra tác động tiêu cực lớn lên cơ sở dữ liệu. Từ đó khóa luận đề xuất thuật toán SO2DI một phương pháp mới kết hợp chiến lược xóa item và tối ưu hóa ngẫu nhiên nhằm bảo vệ tính riêng tư.

Cụ thể hơn, thuật toán sẽ lần lượt duyệt qua từng mẫu hữu ích nhưng nhạy cảm. Với mỗi mẫu này, thuật toán tiến hành

B1: kiểm tra mẫu này đã ẩn hay chưa

B2: Chọn item mục tiêu là các item có tần suất thấp nhất trong các item có lợi ích cao nhưng không nhạy cảm.

B3: Chọn transaction mục tiêu bằng cách tối ưu hóa hàm mục tiêu với hai tiêu chí (1) giảm lợi ích của mẫu xuống dưới ngưỡng lợi ích tối thiểu và (2) giảm thiểu số lượng item lợi ích cao nhưng không nhạy cảm bị ẩn nhầm

B4: Làm nhiễu dữ liệu bằng cách xóa item mục tiêu khỏi các transaction mục tiêu.

B5: cập nhật lại lợi ích của các NSHUI liên quan.

3. Đóng góp Khoa học và thực tiễn:

Khóa luận tiến hành so sánh SO2DI với các phương pháp PPUM phổ biến như MSU-MAU,

MSU-MIU, FILP và PPUMGAT trên sáu tập dữ liệu thực tế (chess, connect, foodmart, mushrooms, pumsb, retail) và hai tập dữ liệu tổng hợp (t20i6d100k, t25i10d10k)

Kết quả thực nghiệm cho thấy SO2DI không chỉ có hiệu suất vượt trội so với các phương pháp trước đó, đặc biệt trên các tập dữ liệu lớn và dày, mà còn giúp giảm thiểu đáng kể các tác dụng phụ trong quá trình bảo vệ thông

4. Báo cáo viết:

Gồm 6 chương, 82 trang. Có nội dung, bố cục hợp lý. Trình bày rõ ràng ý tưởng.

5. Trình bày trước hội đồng:

Tốt.....

6. Công bố khoa học/ ứng dụng thực tế:

Không có.

Đánh giá xếp loại: đạt yêu cầu khóa luận tốt nghiệp bậc ĐH.....

TP.HCM, ngày 11 tháng 03 năm 2025

Giảng viên phản biện



Phạm Trung Nghĩa

Lời cảm ơn

Trước hết, em xin gửi lời cảm ơn chân thành đến Thầy ThS. Nguyễn Ngọc Đức, người đã tận tình hướng dẫn và hỗ trợ em trong suốt quá trình thực hiện khóa luận này. Sự chỉ bảo của Thầy đã giúp em hiểu sâu hơn về vấn đề nghiên cứu và hoàn thành bài báo cáo một cách tốt nhất. Em cũng xin cảm ơn quý Thầy Cô trong khoa Công nghệ Thông tin của Trường Đại học Khoa học Tự Nhiên đã truyền đạt những kiến thức quý báu trong suốt thời gian em học tập tại trường. Những kiến thức đó đã giúp em có nền tảng vững chắc để áp dụng vào thực tế và hoàn thành khóa luận này.

Cuối cùng, do kiến thức và kinh nghiệm còn hạn chế, bài khóa luận chắc chắn không tránh khỏi thiếu sót. Em mong nhận được những góp ý từ Thầy Cô để có thể hoàn thiện hơn và nâng cao kiến thức của mình.

Em xin chân thành cảm ơn!



fit@hcmus

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

ĐỀ CƯƠNG KHÓA LUẬN TỐT NGHIỆP

**BẢO VỆ TÍNH RIÊNG TƯ TRONG KHAI
THÁC MẪU DỰA TRÊN PHƯƠNG PHÁP
TỐI ƯU NGẪU NHIÊN**

(*Privacy-preserving in utility mining with stochastic optimization*)

1 THÔNG TIN CHUNG

Người hướng dẫn:

- ThS. Nguyễn Ngọc Đức (Khoa Công nghệ Thông tin)

[Nhóm] Sinh viên thực hiện:

1. Trần Khắc Bình (MSSV: 20120437)

Loại đề tài: Nghiên cứu

Thời gian thực hiện: Từ 9/2024 đến 3/2025

2 NỘI DUNG THỰC HIỆN

2.1 Giới thiệu về đề tài

Trong thời đại dữ liệu lớn, khám phá tri thức từ cơ sở dữ liệu được ứng dụng thực tế trong nhiều lĩnh vực khác nhau. Một trong những kỹ thuật nổi bật trong lĩnh vực này là khai thác mẫu hữu ích (High-Utility Pattern Mining - HUPM), cho phép phát hiện các tri thức hữu ích và các mối tương quan thú vị tiềm ẩn trong dữ liệu. Tuy nhiên, quá trình khai thác này đang đối mặt với một thách thức lớn đó là bảo vệ tính riêng tư, khi các kỹ thuật HUPM có thể vô tình tiết lộ những thông tin nhạy cảm. Để giải quyết vấn đề này, lĩnh vực bảo vệ tính riêng tư trong khai thác mẫu hữu ích (Privacy-Preserving Utility Mining - PPUM) đã ra đời, với mục tiêu ẩn các thông tin nhạy cảm trước khi chia sẻ và khai thác dữ liệu.

PPUM là quá trình che giấu các itemset lợi ích cao nhạy cảm (Sensitive High-Utility Itemsets - SHUI) bằng cách làm nhiễu cơ sở dữ liệu, đồng thời duy trì chất lượng của các mẫu hữu ích khác. Tuy nhiên, việc này không hề đơn giản, vì việc làm nhiễu không hiệu quả có thể phá vỡ cấu trúc dữ liệu ban đầu, dẫn đến các tác dụng phụ không mong muốn như che giấu các mẫu không nhạy cảm (Non-Sensitive High-Utility Itemsets - NSHUI) hoặc sinh ra các mẫu nhân tạo không tồn tại trong dữ liệu gốc. Mặc dù các giải pháp PPUM hiện tại đã đạt được thành công trong việc ẩn thông tin nhạy cảm, chúng vẫn gặp hạn chế về hiệu suất xử lý và có thể gây tác động tiêu cực lên cơ sở dữ liệu.

Trong khi đó, các kỹ thuật tối ưu hóa ngẫu nhiên (Stochastic Optimization) đã chứng minh được hiệu quả vượt trội trong việc giải quyết các bài toán tối ưu phức tạp với thời gian tính toán giới hạn. Xuất phát từ tiềm năng này, khóa luận đề xuất một hướng tiếp cận mới dựa trên tối ưu hóa ngẫu nhiên nhằm khắc phục những hạn chế của các phương pháp PPUM hiện tại, từ đó nâng cao hiệu quả bảo vệ tính riêng tư và tối ưu hóa giá trị khai thác từ dữ liệu.

2.2 Mục tiêu đề tài

Trong kỷ nguyên số, dữ liệu được xem như một tài sản quý giá, mang lại lợi ích to lớn cho các tổ chức và doanh nghiệp thông qua việc khai thác các mẫu dữ liệu có lợi ích cao. Những thông tin này không chỉ giúp đưa ra các quyết định kinh doanh sáng suốt mà còn hỗ trợ điều chỉnh chiến lược một cách hiệu quả. Tuy nhiên, bên cạnh những cơ hội vượt trội, hoạt động khai thác dữ liệu cũng đặt ra thách thức lớn về bảo mật thông tin, khi các thông tin nhạy cảm có nguy cơ bị tiết lộ ngoài ý muốn, dẫn đến vi phạm quyền riêng tư của cá nhân hoặc tổ chức. Thực tế, các giải pháp hiện nay trong lĩnh vực bảo vệ tính riêng tư trong khai thác mẫu hữu ích vẫn chưa đạt được sự cân bằng giữa việc bảo vệ tính riêng tư và duy trì chất lượng của các mẫu khai thác. Điều này tạo ra động lực mạnh mẽ để nghiên cứu và phát triển các phương pháp tiên tiến hơn, nhằm giải quyết vấn đề này một cách hiệu quả. Qua đó, không chỉ giảm thiểu rủi ro mà còn nâng cao giá trị ứng dụng thực tiễn của dữ liệu.

2.3 Phạm vi của đề tài

Nội dung nghiên cứu chính của khóa luận tập trung vào việc bảo vệ tính riêng tư trong khai thác mẫu hữu ích. Các đối tượng nghiên cứu trong khóa luận bao gồm: các kỹ thuật khai thác mẫu lợi ích cao trên cơ sở dữ liệu transaction, các phương pháp bảo vệ tính riêng tư trong khai thác mẫu hữu ích, và các thuật toán tối ưu hóa ngẫu nhiên. Trong khóa luận, một thuật toán mới dựa trên phương pháp tối ưu hóa ngẫu nhiên nhằm bảo vệ tính riêng tư trong khai thác mẫu hữu ích sẽ được đề xuất. Các bộ dữ liệu giao dịch thực tế hoặc tổng hợp, như chess, connect, foodmart, mushrooms, pumsb, retail, t20i6d100k, t25i10d10k, ... sẽ được sử dụng để so sánh và đánh giá hiệu quả của thuật toán đề xuất so với các thuật toán hiện có.

2.4 Cách tiếp cận dự kiến

Hai thuật toán PPUM được giới thiệu lần đầu tiên bởi Yeh và Hsu (2010) [1], có tên là HHUIF và MSICF. Trong cả hai thuật toán, số lượng của các item nạn nhân được giảm cho đến khi giá trị lợi ích của các SHUI giảm xuống dưới ngưỡng tiện ích tối thiểu. Sự khác biệt giữa hai thuật toán nằm ở chiến lược chọn item nạn nhân: HHUIF chọn item có lợi ích cao nhất, trong khi MSICF chọn item xuất hiện nhiều nhất trong các itemset nhạy cảm. Về kết quả, cả hai thuật toán đều có thời gian thực thi lớn trên cơ sở dữ liệu lớn do phải quét dữ liệu nhiều lần và gây mất mát một lượng lớn các itemset hữu ích không nhạy cảm.

Nhằm cải thiện hiệu suất của HHUIF, Yun và Kim (2015) [2] đã đề xuất thuật toán FPUTT. Thuật toán này sử dụng cấu trúc cây để giảm số lần quét cơ sở dữ liệu xuống còn ba lần, giúp tăng tốc đáng kể so với HHUIF. Tuy nhiên, do vẫn giữ nguyên chiến lược chọn item nạn nhân của HHUIF, FPUTT tiếp tục gặp phải các tác dụng phụ tương tự. Bên cạnh đó, cây FPUTT đòi hỏi không gian bộ nhớ lớn và chi phí tính toán cao.

Lin và cộng sự (2016) [3] đã giới thiệu hai thuật toán mới là MSU-MAU và MSU-MIU để giảm tác dụng phụ do ẩn các SHUI. Sự khác biệt giữa hai thuật toán này cũng nằm ở chiến lược chọn item nạn nhân: MSU-MAU chọn item có lợi ích lớn nhất trong transaction nạn nhân, trong khi MSU-MIU chọn item có lợi ích nhỏ nhất. Kết quả thực nghiệm cho thấy cả hai thuật toán không chỉ giảm tác dụng phụ tốt hơn HHUIF và MSICF mà còn cải thiện tốc độ nhờ sử dụng bảng chỉ mục để lưu trữ thông tin các transaction liên quan đến từng itemset nhạy cảm.

Ngoài các thuật toán dựa vào phương pháp heuristic ở trên, Li và cộng sự (2019) [4] lần đầu tiên áp dụng quy hoạch số nguyên tuyến tính (Integer Linear Programming - ILP) để giải quyết vấn đề PPUM, với thuật toán là PPUM-ILP. Ý tưởng là mô hình hóa quá trình ẩn các SHUI thành một bài toán thỏa mãn ràng buộc (Constraints Satisfaction Problem - CSP) và sử dụng cơ chế làm lỏng

ràng buộc để tìm lời giải. Kết quả cho thấy, PPUM-ILP giúp giảm đáng kể các tác dụng phụ nhưng thời gian thực thi không ổn định, đặc biệt có thể kéo dài trên các tập dữ liệu lớn.

Để cải thiện hiệu suất của PPUM-ILP, Nguyen và cộng sự (2022) [5] đã đề xuất thuật toán FILP, sử dụng cấu trúc dữ liệu băm để tăng tốc quá trình lọc các itemset và mô hình hóa bài toán. Nhờ đó, FILP đạt được hiệu suất cao hơn so với PPUM-ILP.

Tiếp nối, Nguyen và cộng sự (2023) [6] đã phát triển thuật toán G-ILP nhằm giảm tác dụng phụ và tối ưu thời gian thực thi trong quá trình làm nhiễu. G-ILP áp dụng lập trình song song trên GPU để rút ngắn thời gian tiền xử lý và xây dựng một mô hình thỏa mãn ràng buộc khác hiệu quả hơn. Kết quả thực nghiệm chứng minh rằng G-ILP vượt trội cả về thời gian thực thi và khả năng giảm thiểu tác dụng phụ, đặc biệt trên các tập dữ liệu lớn và thưa, so với tất cả các thuật toán trước đó.

Tất cả các thuật toán được thảo luận ở trên đều áp dụng mô hình làm nhiễu dựa trên item để ẩn SHUI, trong đó số lượng của các item nạn nhân được sửa đổi. Ngược lại, một số nghiên cứu khác tuân theo mô hình nhiễu dựa trên transaction cho nhiệm vụ PPUM. Tiêu biểu trong số đó là hai thuật toán PPUMGA+insert [7] và PPUMGAT [8] do Lin và cộng sự đề xuất, cả hai đều dựa trên thuật toán di truyền (Genetic Algorithm - GA).

PPUMGA+insert là thuật toán đầu tiên theo mô hình nhiễu dựa trên transaction, được Lin và cộng sự giới thiệu vào năm 2014. Thuật toán hoạt động bằng cách chèn thêm các transaction mới vào cơ sở dữ liệu nhằm tăng tổng lợi ích của cơ sở dữ liệu. Khi tổng lợi ích tăng, những lợi ích tối thiểu cũng được nâng lên, từ đó làm cho các SHUI không còn được coi là lợi ích cao. Tuy nhiên, việc thêm các transaction mới có thể làm thay đổi cấu trúc dữ liệu và dẫn đến sự xuất hiện của các itemset hữu ích nhân tạo trong cơ sở dữ liệu sau khi làm nhiễu.

Tiếp nối, vào năm 2017, Lin và cộng sự đã phát triển một thuật toán khác có tên

PPUMGAT. Thuật toán này sử dụng cách tiếp cận đối lập so với PPUMGA+insert: thay vì thêm transaction, PPUMGAT thực hiện xóa các transaction hiện có trong cơ sở dữ liệu nhằm giảm lợi ích của SHUI. PPUMGAT sử dụng hàm mục tiêu để đánh giá mức độ phù hợp của các giải pháp ứng cử viên, từ đó xác định các transaction nhạy cảm cần xóa. Tuy nhiên, cũng giống như PPUMGA+insert, PPUMGAT phải đổi mới với vấn đề làm thay đổi số lượng transaction trong cơ sở dữ liệu, có thể dẫn đến mất mát thông tin và tạo ra các itemset hữu ích nhân tạo.

Trong khóa luận này, thuật toán đề xuất áp dụng mô hình nhiều dựa trên item để ẩn các SHUI. Tương tự như HHUIF, thuật toán tiến hành xử lý lần lượt từng SHUI để đảm bảo rằng lợi ích của chúng giảm xuống dưới ngưỡng lợi ích tối thiểu. Tuy nhiên, thay vì sửa đổi số lượng các item, thuật toán lựa chọn xóa item trong các transaction nạn nhân, giúp rút ngắn thời gian xử lý và tăng hiệu quả làm nhiều. Item nạn nhân được lựa chọn dựa trên tần suất xuất hiện ít nhất trong các NSHUI. Cách tiếp cận này đảm bảo rằng việc xóa item sẽ ít ảnh hưởng nhất đến các NSHUI, qua đó giảm thiểu nguy cơ làm ẩn các NSHUI sau khi thực hiện quá trình làm nhiều. Hàm mục tiêu được thiết kế để đánh giá mức độ phù hợp của các giải pháp ứng cử viên trong việc xác định các transaction nạn nhân. Hàm này cân nhắc đồng thời hai mục tiêu: đảm bảo SHUI bị ẩn và giảm thiểu số lượng NSHUI bị ẩn nhầm. Phương pháp tối ưu hóa ngẫu nhiên được sử dụng để tối ưu hóa hàm mục tiêu, cho phép tìm kiếm các giải pháp tốt nhất trong không gian tìm kiếm rộng lớn với thời gian xử lý giới hạn.

2.5 Kết quả dự kiến của đề tài

Dưới đây là các kết quả cụ thể mà khóa luận hướng đến đạt được:

- Tìm hiểu vấn đề khai thác mẫu hữu ích.
- Nghiên cứu vấn đề bảo vệ tính riêng tư trong khai thác mẫu hữu ích.

- Nghiên cứu các phương pháp và thuật toán tối ưu hóa ngẫu nhiên.
- Đề xuất một phương pháp hoặc hướng tiếp cận mới nhằm bảo vệ tính riêng tư trong khai thác mẫu hữu ích.
- Cài đặt thuật toán khai thác mẫu hữu ích EFIM.
- Cài đặt các thuật toán bảo vệ tính riêng tư trong khai thác mẫu hữu ích: MSU-MAU, MSU-MIU, FILP và PPUMGAT.
- Xây dựng chương trình thử nghiệm và đánh giá phương pháp đề xuất dựa trên kết quả thực nghiệm.

2.6 Kế hoạch thực hiện

Thời gian	Công việc	Kết quả dự kiến
9/2024	Nghiên cứu tổng quan, tìm hiểu các phương pháp hiện có	Xác định hướng tiếp cận
10/2024	Thiết kế thuật toán	Đề xuất thuật toán chi tiết
11/2024	- Cài đặt thuật toán - Thủ nghiệm ban đầu	- Phiên bản đầu tiên - Kết quả thử nghiệm sơ bộ
12/2024	- Tối ưu hóa thuật toán - Thực nghiệm trên các bộ dữ liệu	- Phiên bản cải tiến - Kết quả thực nghiệm chi tiết
1/2025	- Phân tích kết quả - Viết báo cáo	- Phân tích so sánh - Bản thảo báo cáo
2/2025	- Hoàn thiện báo cáo - Chuẩn bị bảo vệ	- Báo cáo hoàn chỉnh - Slides bảo vệ
3/2025	Chỉnh sửa theo góp ý và bảo vệ khóa luận	Báo cáo cuối cùng

Tài liệu

- [1] J.-S. Yeh and P.-C. Hsu, "Hhuif and msicf: Novel algorithms for privacy preserving utility mining," *Expert Syst. Appl.*, vol. 37, no. 7, pp. 4779–4786, 2010.
- [2] U. Yun and J. Kim, "A fast perturbation algorithm using tree structure for privacy preserving utility mining," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1149–1165, 2015.
- [3] J. C.-W. Lin, T.-Y. Wu, P. Fournier-Viger, G. Lin, J. Zhan, and M. Voznak, "Fast algorithms for hiding sensitive high-utility itemsets in privacy-preserving utility mining," *Eng. Appl. Artif. Intell.*, vol. 55, pp. 269–284, 2016.
- [4] S. Li, M. Nankun, J. Le, and X. Liao, "A novel algorithm for privacy preserving utility mining based on integer linear programming," *Eng. Appl. Artif. Intell.*, vol. 81, pp. 300–312, 2019.
- [5] D. Nguyen and B. Le, "A fast algorithm for privacy-preserving utility mining," *J. Inf. Technol. Commun.*, vol. 2022, no. 1, pp. 12–22, 2022.
- [6] D. Nguyen, M.-T. Tran, and B. Le, "A new algorithm using integer programming relaxation for privacy-preserving in utility mining," *Applied Intelligence*, vol. 2023, pp. Article ID 10489–023–04913–w, 2023.
- [7] C.-W. Lin, T.-P. Hong, J.-W. Wong, G.-C. Lan, and W.-Y. Lin, "A ga-based approach to hide sensitive high utility itemsets," *Sci. World J.*, vol. 2014, pp. 1–12, 2014.
- [8] J.-W. Lin, T.-P. Hong, P. Fournier-Viger, Q. Liu, J.-W. Wong, and J. Zhan, "Efficient hiding of confidential high-utility itemsets with minimal side effects," *J. Exp. Theor. Artif. Intell.*, vol. 132, p. 103360, 2017.

XÁC NHẬN
CỦA NGƯỜI HƯỚNG DẪN
(Ký và ghi rõ họ tên)

QĐ
Nguyễn Ngọc Đức

TP. Hồ Chí Minh, ngày 20 tháng 2 năm 2025
NHÓM SINH VIÊN THỰC HIỆN
(Ký và ghi rõ họ tên)

Binh
Trần Khắc Bình

Mục lục

Lời cam đoan	i
Nhận xét của GV hướng dẫn	ii
Nhận xét của GV phản biện	iv
Lời cảm ơn	vi
Đề cương	vii
Mục lục	xvi
Danh sách hình	xix
Danh sách bảng	xxi
Danh sách chữ viết tắt	xxi
Tóm tắt	xxiv
1 Mở đầu	1
1.1 Động lực nghiên cứu	1
1.2 Nội dung và phạm vi nghiên cứu	2
1.3 Bố cục của luận văn	3
2 Kiến thức nền tảng	4

2.1	Định nghĩa	4
2.2	Phát biểu bài toán	8
3	Các công trình liên quan	12
3.1	Khai thác mẫu hữu ích	12
3.2	Thuật toán khai thác mẫu hữu ích EFIM	15
3.2.1	Định nghĩa	15
3.2.2	Không gian tìm kiếm	16
3.2.3	Giảm chi phí quét cơ sở dữ liệu bằng phép chiếu . .	17
3.2.4	Giảm chi phí quét cơ sở dữ liệu bằng hợp nhất transaction	18
3.2.5	Cắt tỉa không gian tìm kiếm bằng Sub-tree Utility and Local Utility	19
3.2.6	Tính giới hạn trên hiệu quả bằng Fast Utility Counting	20
3.2.7	Thuật toán EFIM	21
3.3	Bảo vệ tính riêng tư trong khai thác mẫu hữu ích	23
3.4	Một số thuật toán bảo vệ tính riêng tư trong khai thác mẫu hữu ích	27
3.4.1	MSU-MAU và MSU-MIU	27
3.4.2	FILP	29
3.4.3	PPUMGAT	34
3.5	Tối ưu hóa ngẫu nhiên	36
3.6	Một số thuật toán tối ưu hóa ngẫu nhiên	40
3.6.1	Genetic Algorithm	40
3.6.2	Particle Swarm Optimization	44
3.6.3	Ant Colony Optimization	47
4	Thuật toán đề xuất SO2DI	50
4.1	Kiểm tra xem SHUI đã bị ăn hay chưa	51
4.2	Xác định item mục tiêu	52
4.3	Xác định transaction mục tiêu	53
4.3.1	Mô hình hóa bài toán	54

4.3.2	Hàm mục tiêu	54
4.3.3	Xác định số lượng transaction mục tiêu tối đa . . .	57
4.3.4	Giải bài toán bằng phương pháp tối ưu hóa ngẫu nhiên	58
4.4	Làm nhiễu cơ sở dữ liệu	59
4.5	Cập nhật lợi ích cho các NSHUI	59
4.6	Ví dụ minh họa	62
5	Thực nghiệm và đánh giá	65
5.1	Môi trường thực nghiệm và dữ liệu sử dụng	65
5.2	Kết quả thực nghiệm	67
5.2.1	Thời gian thực thi	67
5.2.2	Tác dụng phụ	71
6	Kết luận và hướng phát triển	80
6.1	Kết luận	80
6.2	Hướng phát triển	82
Tài liệu tham khảo		83

Danh sách hình

2.1	Hình minh họa quy trình PPUM [1].	9
2.2	Mối quan hệ giữa I_H , I_S , I_N và I'_H trong PPUM [3].	11
3.1	Cây liệt kê tập hợp cho $I = \{a, b, c, d\}$ [10].	17
3.2	Một ví dụ về biểu diễn di truyền trong GA.	41
3.3	Một số phương pháp lai ghép phổ biến trong GA.	42
3.4	Hình minh họa quá trình đột biến trong GA.	42
3.5	Lưu đồ thuật toán GA.	43
3.6	Lưu đồ thuật toán PSO.	46
3.7	Lưu đồ thuật toán ACO.	49
5.1	So sánh thời gian thực thi trên các tập dữ liệu nhỏ với các ngưỡng lợi ích tối thiểu khác nhau.	67
5.2	So sánh thời gian thực thi trên các tập dữ liệu lớn với các ngưỡng lợi ích tối thiểu khác nhau.	68
5.3	So sánh thời gian thực thi trên các tập dữ liệu nhỏ với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.	69
5.4	So sánh thời gian thực thi trên các tập dữ liệu lớn với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.	70
5.5	So sánh Hiding Failure của các thuật toán với các ngưỡng lợi ích tối thiểu khác nhau.	71
5.6	So sánh Hiding Failure của các thuật toán với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.	72

5.7	So sánh Missing Cost trên các tập dữ liệu nhỏ với các ngưỡng lợi ích tối thiểu khác nhau.	74
5.8	So sánh Missing Cost trên các tập dữ liệu lớn với các ngưỡng lợi ích tối thiểu khác nhau.	75
5.9	So sánh Missing Cost trên các tập dữ liệu nhỏ với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.	76
5.10	So sánh Missing Cost trên các tập dữ liệu lớn với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.	77
5.11	So sánh Artificial Cost của các thuật toán với các ngưỡng lợi ích tối thiểu khác nhau.	78
5.12	So sánh Artificial Cost của các thuật toán với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.	79

Danh sách bảng

2.1	Cơ sở dữ liệu định lượng	5
2.2	Bảng lợi ích ngoại	5
2.3	Bảng HUIs	8
3.1	Phân loại các thuật toán PPUM.	26
4.1	Bảng NSHUIs	62
4.2	Bảng NSHUIs sau khi ẩn $\{AC\}$	63
4.3	Bảng NSHUIs sau khi ẩn $\{AC\}, \{BCE\}$	63
4.4	Cơ sở dữ liệu nhiễu D'	64
5.1	Đặc điểm của các tập dữ liệu thực nghiệm	66

Danh sách chữ viết tắt

Ký hiệu	Ý nghĩa
ACO	Tối ưu đàm kiến (Ant Colony Optimization)
CSP	Bài toán thỏa mãn ràng buộc (Constraints Satisfaction Problem)
EU	Lợi ích ngoại (External Utility)
FPM	Khai thác mẫu phổ biến (Frequent Pattern Mining)
GA	Thuật toán di truyền (Genetic Algorithm)
HUI	Itemset lợi ích cao (High-Utility Itemset)
HUPM	Khai thác mẫu hữu ích (High-Utility Pattern Mining)
ILP	Quy hoạch số nguyên tuyến tính (Integer Linear Programming)
IU	Lợi ích nội (Internal Utility)

Ký hiệu	Ý nghĩa
NSHUI	Itemset lợi ích cao không nhạy cảm (Non-Sensitive High-Utility Itemset)
PPUM	Bảo vệ tính riêng tư trong khai thác mảng hữu ích (Privacy-Preserving Utility Mining)
PSO	Tối ưu bầy đàn hạt (Particle Swarm Optimization)
SHUI	Itemset lợi ích cao nhạy cảm (Sensitive High-Utility Itemset)
TWU	Lợi ích có trọng số transaction (Transaction Weighted Utility)

Tóm tắt

Trong bối cảnh dữ liệu lớn, khai thác mẫu hữu ích (HUPM) đã trở thành một công cụ quan trọng để khám phá tri thức từ dữ liệu. Tuy nhiên, quá trình khai thác này có thể vô tình tiết lộ thông tin nhạy cảm, gây ra những rủi ro về quyền riêng tư. Do đó, lĩnh vực bảo vệ tính riêng tư trong khai thác mẫu hữu ích (PPUM) ra đời nhằm che giấu các mẫu hữu ích nhưng nhạy cảm trước khi dữ liệu được công bố hoặc sử dụng. Một thách thức lớn của PPUM là sau quá trình bảo vệ, dữ liệu có thể bị biến dạng, làm giảm chất lượng khai thác và ảnh hưởng đến hiệu quả của HUPM.

Xuất phát từ vấn đề này, luận văn đã nghiên cứu và đề xuất một thuật toán PPUM mới có tên là SO2DI, kết hợp tối ưu hóa ngẫu nhiên với chiến lược xóa item để làm nhiễu dữ liệu hiệu quả, đảm bảo che giấu các mẫu nhạy cảm mà vẫn duy trì tính hữu ích của dữ liệu. Để đánh giá hiệu suất của thuật toán, luận văn tiến hành so sánh SO2DI với các phương pháp PPUM phổ biến như MSU-MAU, MSU-MIU, FILP và PPUMGAT trên nhiều tập dữ liệu khác nhau. Kết quả thực nghiệm cho thấy SO2DI có hiệu suất và khả năng giảm thiểu tác dụng phụ vượt trội so với các phương pháp trước đó, đặc biệt trên các tập dữ liệu lớn và có mật độ cao.

Luận văn không chỉ đóng góp một phương pháp hiệu quả cho bài toán PPUM, mà còn mở ra hướng nghiên cứu mới trong việc ứng dụng các thuật toán tối ưu hóa ngẫu nhiên vào lĩnh vực này trong tương lai.

Chương 1

Mở đầu

Chương này trình bày động lực nghiên cứu của luận văn, xuất phát từ nhu cầu bảo vệ tính riêng tư trong khai thác mẫu hữu ích. Đồng thời, chương này xác định nội dung và phạm vi nghiên cứu của luận văn, bao gồm việc nghiên cứu các kỹ thuật khai thác mẫu hữu ích, các phương pháp bảo vệ tính riêng tư và các thuật toán tối ưu hóa ngẫu nhiên. Cuối cùng, bô cục của luận văn được giới thiệu để giúp người đọc có cái nhìn tổng quan về toàn bộ nội dung và cách tổ chức của luận văn.

1.1 Động lực nghiên cứu

Trong thời đại dữ liệu lớn, khám phá tri thức từ dữ liệu được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau. Một trong những kỹ thuật nổi bật trong lĩnh vực này là khai thác mẫu hữu ích (High-Utility Pattern Mining - HUPM), cho phép phát hiện các tri thức có giá trị và các mối tương quan thú vị tiềm ẩn trong dữ liệu. Tuy nhiên, khi triển khai HUPM, rủi ro về tiết lộ thông tin nhạy cảm luôn tồn tại, gây ra những lo ngại nghiêm trọng về bảo vệ tính riêng tư.

Để giải quyết vấn đề này, lĩnh vực bảo vệ tính riêng tư trong khai thác mẫu hữu ích (Privacy-Preserving Utility Mining - PPUM) đã ra đời, với mục tiêu ẩn các thông tin nhạy cảm trước khi dữ liệu được chia sẻ và

khai thác. PPUM thực hiện che giấu các mẫu hữu ích nhưng nhạy cảm (Sensitive High-Utility Itemsets - SHUI) thông qua việc làm nhiễu cơ sở dữ liệu, đồng thời cố gắng duy trì chất lượng của các mẫu hữu ích còn lại. Tuy nhiên, nếu quá trình làm nhiễu không hiệu quả, nó có thể làm biến dạng cấu trúc dữ liệu ban đầu, dẫn đến các tác dụng phụ không mong muốn như ẩn đi các mẫu không nhạy cảm (Non-Sensitive High-Utility Itemsets - NSHUI) hoặc sinh ra các mẫu nhân tạo không tồn tại trong dữ liệu gốc.

Mặc dù một số giải pháp PPUM hiện nay đã đạt được thành công trong việc bảo vệ thông tin nhạy cảm, chúng vẫn gặp hạn chế về hiệu suất xử lý và có thể gây ra tác động tiêu cực lớn lên cơ sở dữ liệu. Điều này tạo ra động lực mạnh mẽ để tiếp tục nghiên cứu và phát triển các phương pháp tiên tiến hơn, nhằm giải quyết hiệu quả vấn đề bảo vệ thông tin nhạy cảm trong khi vẫn giữ được giá trị khai thác của dữ liệu. Qua đó, không chỉ giảm thiểu rủi ro mà còn nâng cao giá trị ứng dụng thực tiễn của dữ liệu.

1.2 Nội dung và phạm vi nghiên cứu

Nội dung nghiên cứu chính của khóa luận tập trung vào việc bảo vệ tính riêng tư trong khai thác mẫu hữu ích. Các đối tượng nghiên cứu trong khóa luận bao gồm: các kỹ thuật khai thác mẫu lợi ích cao trên cơ sở dữ liệu transaction, các phương pháp bảo vệ tính riêng tư trong khai thác mẫu hữu ích, và các thuật toán tối ưu hóa ngẫu nhiên. Trong khóa luận, một thuật toán mới dựa trên phương pháp tối ưu hóa ngẫu nhiên nhằm bảo vệ tính riêng tư trong khai thác mẫu hữu ích sẽ được đề xuất. Ngoài ra, các bộ dữ liệu transaction thực tế hoặc tổng hợp, như foodmart, t25i10d10k, mushrooms, chess, retail, t20i6d100k, pumsb, connect, ... sẽ được sử dụng để so sánh và đánh giá hiệu quả của thuật toán đề xuất so với các thuật toán hiện có.

1.3 Bố cục của luận văn

Luận văn được trình bày trong sáu chương, với cấu trúc như sau:

- **Chương 1 - Mở đầu:** Giới thiệu động lực nghiên cứu, xác định nội dung và phạm vi nghiên cứu, đồng thời trình bày tổng quan về bố cục luận văn.
- **Chương 2 - Kiến thức nền tảng:** Cung cấp các khái niệm và định nghĩa cơ bản liên quan đến khai thác mẫu hữu ích và bảo vệ tính riêng tư trong khai thác mẫu hữu ích, đồng thời trình bày các tác dụng phụ của quá trình bảo vệ dữ liệu.
- **Chương 3 - Các công trình liên quan:** Tổng quan về các nghiên cứu và thuật toán trước đây trong lĩnh vực khai thác mẫu hữu ích, bảo vệ tính riêng tư trong khai thác mẫu hữu ích, và tối ưu hóa ngẫu nhiên.
- **Chương 4 - Thuật toán đề xuất SO2DI:** Trình bày chi tiết về thuật toán SO2DI do luận văn đề xuất, bao gồm cách tiếp cận, các bước thực hiện và ví dụ minh họa.
- **Chương 5 - Thực nghiệm và đánh giá:** Đánh giá hiệu suất của thuật toán SO2DI thông qua thực nghiệm trên nhiều tập dữ liệu khác nhau, so sánh với các thuật toán hiện có và phân tích kết quả thực nghiệm.
- **Chương 6 - Kết luận và hướng phát triển:** Tổng kết các kết quả đạt được của luận văn và đề xuất các hướng nghiên cứu trong tương lai nhằm cải thiện thuật toán đề xuất.

Chương 2

Kiến thức nền tảng

Chương này trình bày các khái niệm và định nghĩa cơ bản liên quan đến khai thác mẫu hữu ích (HUPM) và bảo vệ tính riêng tư trong khai thác mẫu hữu ích (PPUM). Trước tiên, các khái niệm về cơ sở dữ liệu transaction định lượng, lợi ích nội, lợi ích ngoai và cách tính giá trị lợi ích của một itemset được giới thiệu nhằm cung cấp nền tảng lý thuyết cho HUPM. Tiếp theo, chương phát biểu bài toán PPUM, làm rõ các thách thức chính và giới thiệu các thước đo Hiding Failure, Missing Cost và Artificial Cost để đánh giá tác dụng phụ của quá trình bảo vệ dữ liệu. Những nội dung này giúp xây dựng cơ sở lý thuyết vững chắc cho các chương tiếp theo.

2.1 Định nghĩa

Một cơ sở dữ liệu transaction định lượng D bao gồm N transaction, ký hiệu $D = \{T_1, T_2, \dots, T_N\}$. Mỗi transaction $T_n \in D (1 \leq n \leq N)$ chứa một hoặc nhiều item từ tập hợp $I = \{i_1, i_2, \dots, i_M\}$, là tập hợp hữu hạn gồm M item khác nhau, và $T_n \subseteq I$. Mỗi transaction T_n có một mã định danh duy nhất gọi là TID (Transaction ID). Một ví dụ về cơ sở dữ liệu transaction định lượng được trình bày ở Bảng 2.1.

Bảng 2.1: Cơ sở dữ liệu định lượng

TID	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
T_1	5	1	6	0	8
T_2	6	6	7	8	3
T_3	8	1	4	0	0
T_4	4	0	0	8	4
T_5	0	1	0	0	0
T_6	9	7	0	9	0
T_7	5	0	3	6	5
T_8	0	6	8	8	9

Định nghĩa 1: Lợi ích nội (Internal Utility)

Lợi ích nội của item i_m trong transaction T_n , ký hiệu $q(i_m, T_n)$, là số lượng hoặc tần suất xuất hiện của item i_m trong transaction T_n .

Ví dụ: Trong Bảng 2.1, lợi ích nội của item *A* trong transaction T_1 là 5, nghĩa là $q(A, T_1) = 5$.

Định nghĩa 2: Lợi ích ngoại (External Utility)

Lợi ích ngoại của item i_m , ký hiệu $p(i_m)$, thể hiện tầm quan trọng, lợi nhuận hoặc trọng số của item i_m trong cơ sở dữ liệu transaction định lượng.

Ví dụ: Trong Bảng 2.2, lợi ích ngoại của item *A* là 4, tức là $p(A) = 4$.

Bảng 2.2: Bảng lợi ích ngoại

Item	External Utility
<i>A</i>	4
<i>B</i>	8
<i>C</i>	8
<i>D</i>	1
<i>E</i>	2

Một cơ sở dữ liệu transaction định lượng thường được sử dụng để mô hình hóa hành vi mua hàng hoặc hoạt động của người dùng, trong đó các item đại diện cho các sản phẩm hoặc hành vi cụ thể. Giá trị internal utility phản ánh số lượng sản phẩm được mua hoặc tần suất hoạt động, trong khi giá trị external utility thể hiện mức độ lợi nhuận hoặc mức độ quan trọng của các sản phẩm đó.

Định nghĩa 3: Lợi ích của item i_m trong transaction T_n

Lợi ích của item i_m trong transaction T_n , ký hiệu là $u(i_m, T_n)$, được tính bằng tích của lợi ích nội và lợi ích ngoại của item i_m :

$$u(i_m, T_n) = q(i_m, T_n) \times p(i_m) \quad (2.1)$$

Ví dụ: Trong Bảng 2.1, lợi ích của item A trong transaction T_1 là: $u(A, T_1) = q(A, T_1) \times p(A) = 5 \times 4 = 20$.

Định nghĩa 4: Itemset

Một k -itemset X là một tập hợp gồm k item khác nhau từ I , ký hiệu $X = \{i_1, i_2, \dots, i_k\}$ với $k \leq M$. Một transaction T_n được coi là hỗ trợ itemset X khi và chỉ khi $X \subseteq T_n$.

Định nghĩa 5: Lợi ích của itemset X trong transaction T_n

Lợi ích của itemset X trong transaction T_n (với $X \subseteq T_n$), ký hiệu là $u(X, T_n)$, được tính bằng tổng lợi ích của tất cả các item thuộc X trong transaction T_n :

$$u(X, T_n) = \sum_{i_m \in X} u(i_m, T_n) \quad (2.2)$$

Nếu X không được hỗ trợ bởi T_n , $u(X, T_n)$ sẽ bằng 0.

Ví dụ: Trong Bảng 2.1, lợi ích của itemset $\{AC\}$ trong transaction T_1 được tính như sau: Vì $\{AC\} \subseteq T_1$, ta có: $u(\{AC\}, T_1) = u(A, T_1) + u(C, T_1) = 5 \times 4 + 6 \times 8 = 68$. Trong khi đó, do $\{AC\} \not\subseteq T_4$, nên $u(\{AC\}, T_4) = 0$.

Định nghĩa 6: Lợi ích của itemset X trong cơ sở dữ liệu D

Lợi ích của itemset X trong cơ sở dữ liệu D , ký hiệu là $u(X)$, được tính bằng tổng lợi ích của X trong tất cả các transaction hỗ trợ X :

$$u(X) = \sum_{T_n \in D, X \subseteq T_n} u(X, T_n) \quad (2.3)$$

Ví dụ: Trong Bảng 2.1, lợi ích của itemset $\{AC\}$ trong cơ sở dữ liệu D được tính như sau: $u(\{AC\}) = u(\{AC\}, T_1) + u(\{AC\}, T_2) + u(\{AC\}, T_3) + u(\{AC\}, T_7) = (5 \times 4 + 6 \times 8) + (6 \times 4 + 7 \times 8) + (8 \times 4 + 4 \times 8) + (5 \times 4 + 3 \times 8) = 68 + 80 + 64 + 44 = 256$.

Định nghĩa 7: Lợi ích của transaction T_n

Lợi ích của transaction T_n , ký hiệu là $tu(T_n)$, được tính bằng tổng lợi ích của tất cả các item trong transaction T_n :

$$tu(T_n) = \sum_{i_m \in T_n} u(i_m, T_n) \quad (2.4)$$

Ví dụ: Trong Bảng 2.1, lợi ích của transaction T_1 là: $tu(T_1) = u(A, T_1) + u(B, T_1) + u(C, T_1) + u(E, T_1) = 5 \times 4 + 1 \times 8 + 6 \times 8 + 8 \times 2 = 92$.

Định nghĩa 8: Tổng lợi ích của cơ sở dữ liệu D

Tổng lợi ích của cơ sở dữ liệu D , ký hiệu là TU , được tính bằng tổng lợi ích của tất cả các transaction trong D :

$$TU = \sum_{T_n \in D} tu(T_n) \quad (2.5)$$

Ví dụ: Trong Bảng 2.1, tổng lợi ích của cơ sở dữ liệu D là: $TU = tu(T_1) + tu(T_2) + tu(T_3) + tu(T_4) + tu(T_5) + tu(T_6) + tu(T_7) + tu(T_8) = 92 + 142 + 72 + 32 + 8 + 101 + 60 + 138 = 645$.

Định nghĩa 9: Ngưỡng lợi ích tối thiểu

Ngưỡng lợi ích tối thiểu, ký hiệu là δ , là một tiêu chí dùng để xác định xem một itemset có phải là itemset lợi ích cao (hữu ích) hay không. Ngưỡng này được biểu diễn dưới dạng một hằng số c , được người dùng chỉ định và không lớn hơn tổng lợi ích của cơ sở dữ liệu, tức là: $\delta = c$, với $0 \leq c \leq TU$.

Định nghĩa 10: Itemset lợi ích cao (High-Utility Itemset - HUI)

Một itemset X được gọi là itemset lợi ích cao nếu lợi ích của nó trong cơ sở dữ liệu D không nhỏ hơn ngưỡng lợi ích tối thiểu δ . Tập hợp các itemset lợi ích cao trong D , ký hiệu là I_H , được định nghĩa như sau:

$$I_H = \{X \subseteq I \mid u(X) \geq \delta\} \quad (2.6)$$

Ví dụ: Trong Bảng 2.1, với ngưỡng lợi ích tối thiểu $\delta = 200$, các itemset lợi ích cao được xác định và liệt kê đầy đủ trong Bảng 2.3.

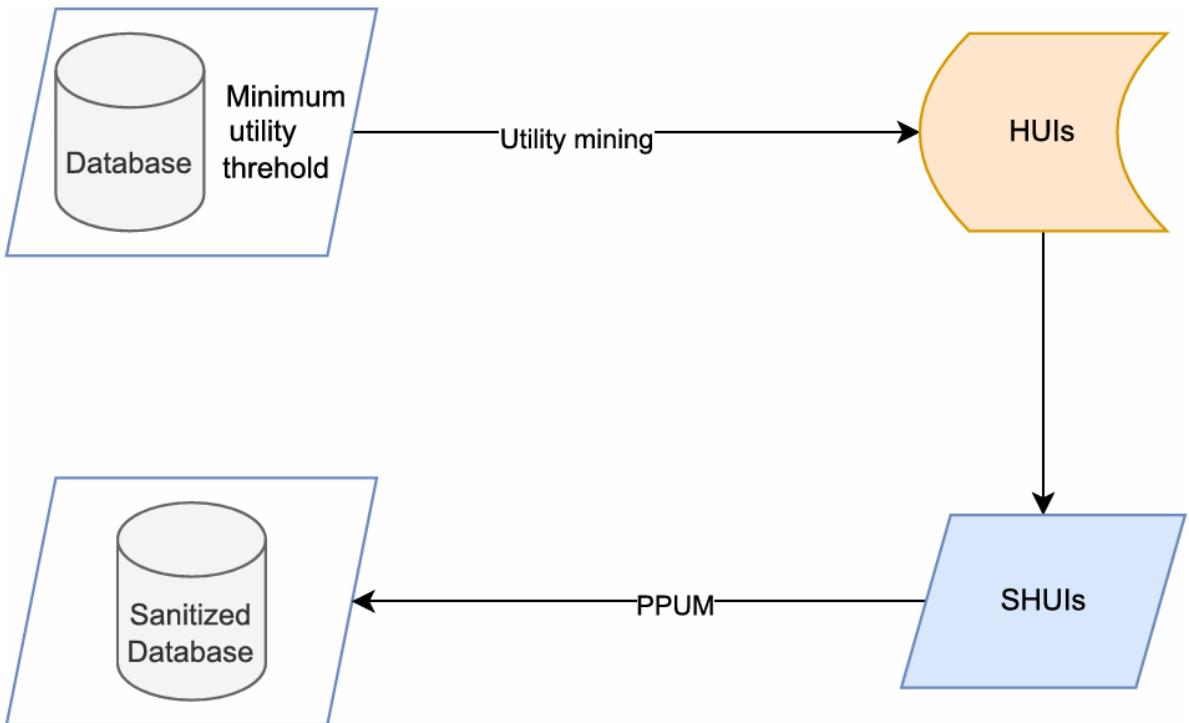
Bảng 2.3: Bảng HUIs

Itemset	Utility	Itemset	Utility	Itemset	Utility
$\{CDE\}$	200	$\{AB\}$	232	$\{AC\}$	256
$\{ACE\}$	224	$\{BCD\}$	232	$\{ABC\}$	276
$\{C\}$	224	$\{CE\}$	242	$\{BCE\}$	312
$\{ABCE\}$	226	$\{BCDE\}$	256	$\{BC\}$	312

2.2 Phát biểu bài toán

Cho một cơ sở dữ liệu transaction định lượng D , tập các itemset lợi ích cao I_H được khai thác dựa trên ngưỡng lợi ích tối thiểu δ . Từ tập I_H , người dùng chỉ định tập các itemset lợi ích cao nhạy cảm I_S , trong đó $I_S \subseteq I_H$. PPUM là quá trình làm nhiễu cơ sở dữ liệu D , biến đổi nó thành cơ sở dữ liệu D' , sao cho tất cả các itemset trong I_S đều bị ẩn giấu. Điều

này có nghĩa rằng, sau khi thực hiện PPUM, với ngưỡng lợi ích tối thiểu δ , không còn itemset nào trong I_S được xem là itemset lợi ích cao trong cơ sở dữ liệu D' . Khung chung của quá trình PPUM có thể được minh họa như trong Hình 2.1 [1].



Hình 2.1: Hình minh họa quy trình PPUM [1].

Thách thức chính của PPUM không chỉ dừng lại ở việc che giấu các itemset nhạy cảm mà còn ở việc giảm thiểu tác động tiêu cực đến chất lượng tổng thể của cơ sở dữ liệu D' . Điều này đòi hỏi việc duy trì tính chính xác và giá trị lợi ích của các itemset không nhạy cảm trong I_H , để cơ sở dữ liệu D' vẫn có thể được sử dụng hiệu quả cho mục đích khai thác mẫu hữu ích.

Để đánh giá các tác dụng phụ do quá trình làm nhiều của PPUM gây ra, Lin và cộng sự [2] đã đề xuất ba thước đo quan trọng. Cụ thể, gọi I_N là tập các itemset lợi ích cao không nhạy cảm trong cơ sở dữ liệu D , được xác định bởi $I_N = I_H - I_S$. Gọi I'_H là tập các itemset lợi ích cao được khai thác từ cơ sở dữ liệu D' .

Định nghĩa 11: Hiding Failure

Hiding Failure, ký hiệu là HF , đo lường tỷ lệ các SHUI mà quá trình làm nhiều không thể che giấu được, tức là tỷ lệ các SHUI trong D vẫn xuất hiện dưới dạng HUI trong D' . Gọi α là số lượng các SHUI không bị ẩn, HF được tính theo công thức sau:

$$HF = \frac{\alpha}{|I_S|} = \frac{|I_S \cap I'_H|}{|I_S|} \quad (2.7)$$

Định nghĩa 12: Missing Cost

Missing Cost, ký hiệu là MC , đo lường tỷ lệ các NSHUI mà quá trình làm nhiều đã vô tình ẩn đi, tức là tỷ lệ các NSHUI trong D không còn xuất hiện dưới dạng HUI trong D' . Gọi β là số lượng các NSHUI bị ẩn nhầm, MC được tính theo công thức sau:

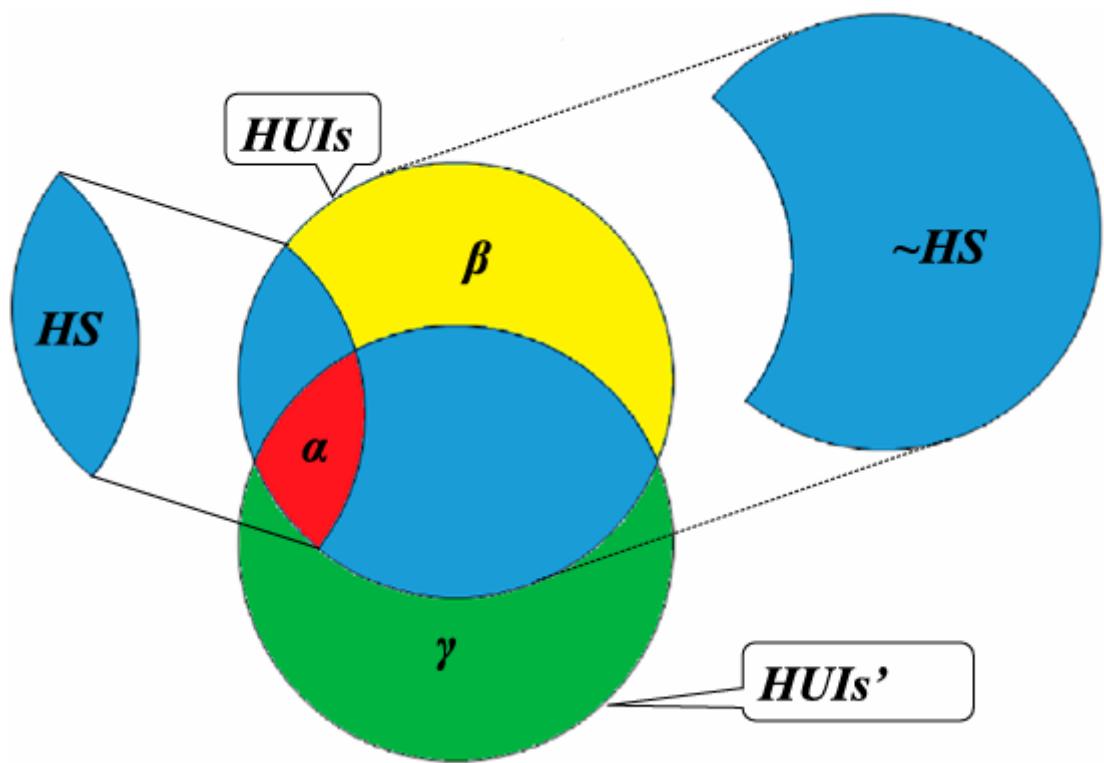
$$MC = \frac{\beta}{|I_N|} = \frac{|I_N - I'_H|}{|I_N|} \quad (2.8)$$

Định nghĩa 13: Artificial Cost

Artificial Cost, ký hiệu là AC , đo lường tỷ lệ các HUI nhân tạo được sinh ra sau quá trình làm nhiều, tức là tỷ lệ các itemset không phải là HUI trong D nhưng lại xuất hiện dưới dạng HUI trong D' . Gọi γ là số lượng các HUI nhân tạo được tạo ra, AC được tính theo công thức sau:

$$AC = \frac{\gamma}{|I_N|} = \frac{|I'_H - I_H|}{|I_N|} \quad (2.9)$$

Hình 2.2 [3] dưới đây biểu diễn mối quan hệ giữa các tập hợp I_H , I_S , I_N và I'_H sau quá trình làm nhiều cơ sở dữ liệu.



Hình 2.2: Mối quan hệ giữa I_H , I_S , I_N và I'_H trong PPUM [3].

Chương 3

Các công trình liên quan

Chương này tổng quan về khai thác mẫu hữu ích (HUPM), bao gồm động lực, ứng dụng và các thuật toán tiêu biểu như Two-Phase, UP-Growth, UP-Growth+, HUI-Miner, FHM và EFIM. Tiếp theo, các nghiên cứu liên quan đến bảo vệ tính riêng tư trong khai thác mẫu hữu ích (PPUM) được phân tích, với các thuật toán đáng chú ý như HHUIF, MSCIF, FPUTT, MSU-MAU, MSU-MIU, PPUM-ILP, FILP, G-ILP, PPUMGA+insert và PPUMGAT. Cuối cùng, chương này giới thiệu tổng quan về tối ưu hóa ngẫu nhiên và một số thuật toán phổ biến như Genetic Algorithm, Particle Swarm Optimization và Ant Colony Optimization, làm cơ sở cho phương pháp đề xuất trong chương tiếp theo.

3.1 Khai thác mẫu hữu ích

Khai thác mẫu hữu ích là một hướng tiếp cận quan trọng trong lĩnh vực khai thác dữ liệu, nhằm tìm ra các itemset có giá trị lợi ích cao trong cơ sở dữ liệu transaction. Lĩnh vực này ra đời nhằm khắc phục hạn chế của khai thác mẫu phổ biến (Frequent Pattern Mining - FPM), vốn chỉ chú trọng đến tần suất xuất hiện của các item mà không đánh giá được giá trị kinh tế của chúng [4]. Ví dụ, trong một cơ sở dữ liệu transaction, các sản phẩm như sữa và bánh mì có thể được mua thường xuyên nhưng không

mang lại lợi nhuận cao, trong khi các sản phẩm cao cấp như rượu vang và pho mát chất lượng có thể mang lại lợi ích kinh tế lớn mặc dù ít xuất hiện. HUPM tổng quát hóa bài toán FPM bằng cách xem xét số lượng và trọng số (giá trị) của các item, từ đó giúp xác định chính xác những itemset có tầm quan trọng thực sự đối với người dùng và doanh nghiệp.

Với khả năng xác định các mẫu có giá trị lợi ích cao, HUPM được ứng dụng rộng rãi trong nhiều lĩnh vực. Trong lĩnh vực bán lẻ, mẫu hữu ích giúp các nhà quản lý tìm ra các sản phẩm mang lại lợi nhuận cao nhất khi được mua chung, từ đó có thể áp dụng các chiến lược tiếp thị và quảng cáo hiệu quả hơn. Trong thương mại điện tử, HUPM hỗ trợ các hệ thống gợi ý sản phẩm bằng cách phân tích không chỉ tần suất mua mà còn cả giá trị giao dịch, giúp cá nhân hóa trải nghiệm mua sắm và tăng hiệu quả bán chéo. Trong lĩnh vực tài chính, HUPM đóng vai trò quan trọng trong phát hiện gian lận, khi các giao dịch có giá trị cao nhưng xuất hiện bất thường có thể là dấu hiệu của hoạt động đáng ngờ, từ đó giúp các tổ chức tài chính nâng cao khả năng bảo mật và giảm thiểu rủi ro.

Mặc dù HUPM mang lại nhiều lợi ích trong việc khai thác thông tin có giá trị, bài toán này lại gặp thách thức lớn do không thỏa mãn thuộc tính đơn điệu (anti-monotonicity) như trong khai thác mẫu phổ biến [4]. Cụ thể, một tập hợp con có thể có giá trị lợi ích thấp, cao hơn hoặc ngang bằng tập hợp cha, do đó không thể áp dụng trực tiếp các chiến lược cắt tỉa không gian tìm kiếm truyền thống. Để khắc phục vấn đề này, các thuật toán HUPM hiện đại đã sử dụng các giới hạn trên nhằm lọc bớt không gian tìm kiếm mà vẫn đảm bảo tìm được tất cả các mẫu hữu ích.

Một trong những bước đột phá ban đầu của bài toán HUPM là thuật toán Two-Phase [5], được phát triển bởi Liu và cộng sự vào năm 2005. Two-Phase kế thừa ý tưởng của thuật toán Apriori, trong đó quá trình khai thác được chia thành hai giai đoạn: giai đoạn đầu sử dụng giới hạn trên TWU (Transaction Weighted Utility) để lọc ra các itemset ứng viên. TWU là một biên trên đơn điệu của giá trị lợi ích, cho phép cắt tỉa không gian tìm kiếm một cách an toàn. Giai đoạn sau tính toán giá trị lợi ích

thực của các itemset ứng viên nhằm loại bỏ các itemset không đạt yêu cầu.

Để khắc phục hạn chế về hiệu suất và số lượng ứng viên lớn của Two-Phase, Tseng và các đồng sự đã đề xuất thuật toán UP-Growth vào năm 2010 [6] và cải tiến sau đó là UP-Growth+ vào năm 2013 [7]. Cả hai đều sử dụng cấu trúc cây tiền tố (prefix-tree) để tổ chức thông tin transaction, kết hợp với phương pháp tìm kiếm theo chiều sâu, qua đó giảm thiểu số lần quét cơ sở dữ liệu và tăng cường hiệu quả loại bỏ các itemset ứng viên không khả thi. UP-Growth+ cải tiến hơn so với UP-Growth ở các kỹ thuật cắt tỉa nhằm giảm biên trên TWU, giúp loại bỏ nhiều ứng viên hơn.

Một bước tiến quan trọng khác trong lĩnh vực HUPM là sự ra đời của thuật toán HUI-Miner [8], do Liu và Qu đề xuất năm 2012. Thuật toán này dựa trên phương pháp duyệt theo chiều sâu kết hợp với khái niệm Utility-List. Với cách tiếp cận này, chỉ cần quét cơ sở dữ liệu một lần để xây dựng utility-list cho các item đơn lẻ. Utility-list cho phép tính toán trực tiếp giá trị lợi ích và biên trên của các itemset, giúp loại bỏ ngay các ứng viên không tiềm năng trong quá trình tìm kiếm.

Năm 2014, Philippe Fournier-Viger và cộng sự đã giới thiệu thuật toán FHM [9], một cải tiến dựa trên HUI-Miner. FHM sử dụng cấu trúc utility-list để lưu trữ thông tin về giá trị lợi ích của các itemset và áp dụng chiến lược duyệt theo chiều sâu để khám phá không gian tìm kiếm. Điểm nổi bật của FHM là việc giới thiệu cấu trúc Estimated Utility Co-occurrence Structure (EUCS) để lưu trữ TWU của các cặp item, từ đó tăng hiệu quả cắt tỉa không gian tìm kiếm. Kết quả thực nghiệm cho thấy FHM nhanh hơn đến sáu lần so với HUI-Miner.

Năm 2015, Souleymane Zida và cộng sự đã giới thiệu thuật toán EFIM [10], nhằm cải thiện hiệu suất khai thác itemset hữu ích. EFIM áp dụng phương pháp duyệt theo chiều sâu và biểu diễn cơ sở dữ liệu theo chiều ngang. Thuật toán này giới thiệu hai biến trên mới là local-utility và subtree-utility, giúp giảm đáng kể không gian tìm kiếm. Ngoài ra, EFIM sử dụng kỹ thuật đếm utility dựa trên mảng cùng các kỹ thuật chiết cơ

sở dữ liệu và hợp nhất transaction để tăng tốc độ xử lý. Thực nghiệm cho thấy EFIM nhanh hơn hai đến ba bậc độ lớn so với các thuật toán trước đó như HUI-Miner, FHM và UP-Growth+.

3.2 Thuật toán khai thác mẫu hữu ích EFIM

Bài toán khai thác mẫu hữu ích có thể được phát biểu như sau:

Cho một cơ sở dữ liệu transaction định lượng D và ngưỡng lợi ích tối thiểu δ . Vấn đề của khai thác mẫu hữu ích là khám phá tất cả các itemset hữu ích - itemset có giá trị lợi ích không nhỏ hơn δ .

3.2.1 Định nghĩa

Định nghĩa 14: Lợi ích có trọng số transaction - TWU

TWU của itemset X , ký hiệu là $twu(X)$, được tính bằng tổng lợi ích transaction của tất cả các transaction chứa itemset X :

$$twu(X) = \sum_{T_n \in D \wedge X \subseteq T_n} tu(T_n) \quad (3.1)$$

Tính chất 1: Cắt tỉa không gian tìm kiếm sử dụng TWU

Với itemset X bất kỳ, nếu $twu(X) < \delta$, thì X là itemset lợi ích thấp và tất cả các itemset cha của nó cũng vậy.

Tính chất này cho phép cắt tỉa không gian tìm kiếm bằng cách loại bỏ sớm các itemset có TWU thấp hơn δ .

Định nghĩa 15: Remaining utility

Đặt \succ là một thứ tự toàn phần trên các item từ I , và X là một itemset. Remaining utility của X trong một transaction T_n được định nghĩa là:

$$re(X, T_n) = \sum_{i \in T_n \wedge i \succ x \forall x \in X} u(i, T_n) \quad (3.2)$$

Định nghĩa 16: Utility-list

Utility-list của một itemset X trong cơ sở dữ liệu D là một tập hợp các bộ sao cho có một bộ $(n, iutil, rutil)$ cho mỗi transaction T_n chứa X . Các phần tử $iutil$ và $rutil$ của một bộ tương ứng là lợi ích của X trong T_n ($u(X, T_n)$) và remaining utility của X trong T_n ($re(X, T_n)$).

Định nghĩa 17: Giới hạn trên của remaining utility

Giả sử X là một itemset. Giả sử các phần mở rộng của X là các itemset có thể thu được bằng cách thêm một item i vào X sao cho $i \succ x, \forall x \in X$. Giới hạn trên remaining utility của X được định nghĩa là:

$$reu(X) = u(X) + re(X) \quad (3.3)$$

Tính chất 2: Cắt tỉa không gian tìm kiếm sử dụng Utility-list

Nếu $reu(X) < \delta$, thì X là một itemset lợi ích thấp và tất cả các itemset mở rộng của nó cũng vậy.

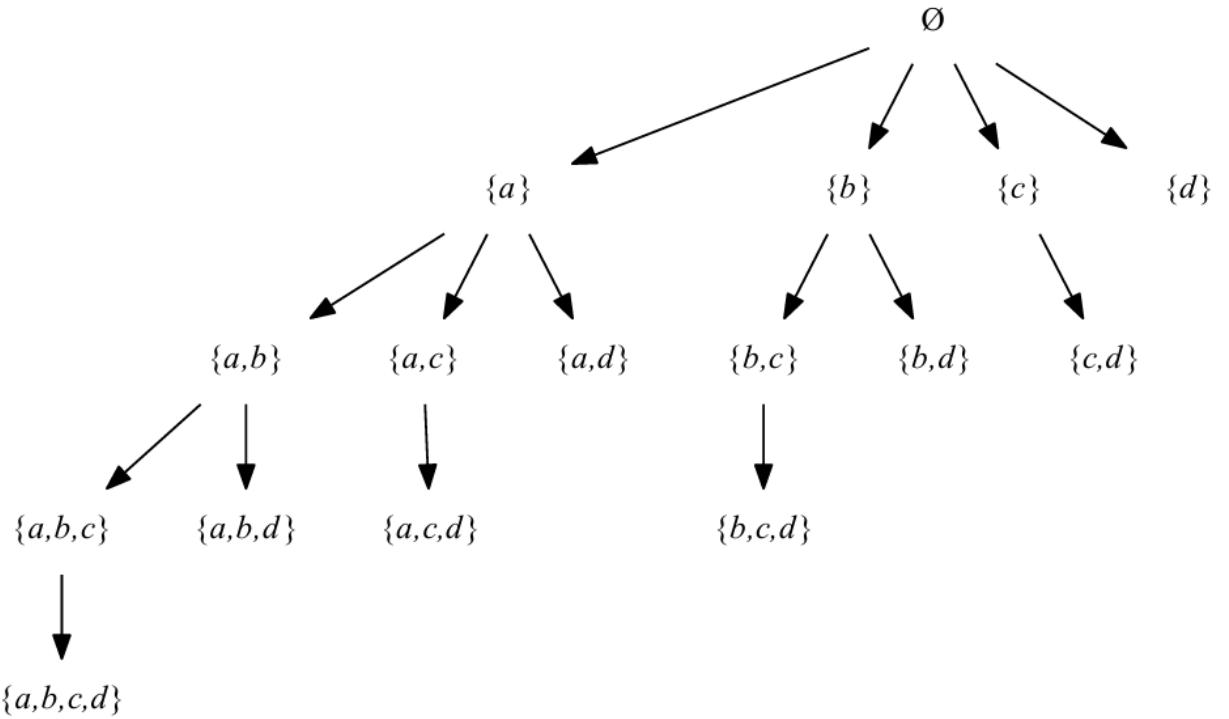
3.2.2 Không gian tìm kiếm

Không gian tìm kiếm trong EFIM được biểu diễn dưới dạng cây liệt kê tập hợp. Cây này được xây dựng dựa trên một thứ tự TWU tăng dần trên các item. Điều này đã được chứng minh trong [7]–[9] là có thể làm giảm không gian tìm kiếm. Thuật toán EFIM sử dụng tìm kiếm theo chiều sâu (depth-first search) để duyệt qua cây liệt kê tập hợp, bắt đầu từ nút gốc (tập rỗng). Hình 3.1 [10] dưới đây minh họa cây liệt kê tập hợp cho tập mục $I = \{a, b, c, d\}$ với giả sử rằng thứ tự từ vựng cũng là thứ tự TWU tăng dần của các item.

Định nghĩa 18: Các item có thể mở rộng một itemset

Giả sử α là một itemset. Đặt $E(\alpha)$ là tập hợp tất cả các item có thể được sử dụng để mở rộng α theo tìm kiếm theo chiều sâu, tức là:

$$E(\alpha) = \{z \mid z \in I \wedge z \succ x, \forall x \in \alpha\} \quad (3.4)$$



Hình 3.1: Cây liệt kê tập hợp cho $I = \{a, b, c, d\}$ [10].

Định nghĩa 19: Mở rộng một itemset.

Giả sử α là một itemset. Một itemset Z là phần mở rộng của α nếu $Z = \alpha \cup W$, với W là tập con khác rỗng của $E(\alpha)$. Ngoài ra, Z được coi là phần mở rộng đơn mục của α nếu $Z = \alpha \cup \{z\}$, với z là một item sao cho $z \in E(\alpha)$.

3.2.3 Giảm chi phí quét cơ sở dữ liệu bằng phép chiếu

Trong quá trình tìm kiếm HUI, EFIM cần phải quét cơ sở dữ liệu nhiều lần để tính toán các giá trị hỗ trợ cho việc đánh giá và cắt tỉa không gian tìm kiếm. Tuy nhiên, việc quét toàn bộ cơ sở dữ liệu cho mỗi lần đánh giá sẽ gây tốn kém về thời gian, đặc biệt là với các cơ sở dữ liệu lớn.

Kỹ thuật chiếu cơ sở dữ liệu (Database Projection) được EFIM sử dụng để giải quyết vấn đề này. Ý tưởng chính của kỹ thuật này là loại bỏ những item không cần thiết khỏi cơ sở dữ liệu trong mỗi bước tìm kiếm. Cụ thể, khi thuật toán đang xem xét một itemset α , các item không thuộc tập

$E(\alpha)$ sẽ không đóng góp vào việc tính toán lợi ích của các itemset con của α . Do đó, ta có thể loại bỏ các item này khỏi cơ sở dữ liệu mà không làm ảnh hưởng đến kết quả của quá trình tìm kiếm.

Định nghĩa 20: Cơ sở dữ liệu chiếu

Phép chiếu của transaction T sử dụng itemset α được ký hiệu là $\alpha - T$. Hình chiếu của cơ sở dữ liệu D sử dụng itemset α được ký hiệu là $\alpha - D$.

$$\alpha - T = \{i \mid i \in T \wedge i \in E(\alpha)\} \quad (3.5)$$

$$\alpha - D = \{\alpha - T \mid T \in D \wedge \alpha - T \neq \emptyset\} \quad (3.6)$$

3.2.4 Giảm chi phí quét cơ sở dữ liệu bằng hợp nhất transaction

Kỹ thuật hợp nhất transaction dựa trên nhận xét rằng các cơ sở dữ liệu transaction thường chứa các transaction giống nhau. Việc gộp các transaction này thành một transaction duy nhất giúp giảm kích thước cơ sở dữ liệu và do đó giảm chi phí quét. Hai transaction được coi là giống nhau nếu chúng chứa cùng một tập các item, không nhất thiết phải có cùng giá trị lợi ích nội. Hợp nhất transaction là quá trình thay thế tập hợp các transaction giống nhau $T_{r_1}, T_{r_2}, \dots, T_{r_m}$ trong cơ sở dữ liệu D bằng một transaction mới duy nhất T_M sao cho số lượng của mỗi item $i \in T_M$ được định nghĩa là tổng số lượng của item đó trong tất cả các transaction giống nhau.

Để xác định các transaction giống nhau một cách hiệu quả, EFIM sắp xếp cơ sở dữ liệu D theo thứ tự \succ_T , được định nghĩa là thứ tự từ điển khi các transaction được đọc ngược. Các transaction giống nhau sẽ xuất hiện liên tiếp trong cơ sở dữ liệu được sắp xếp theo thứ tự \succ_T . Do đó, việc xác định các transaction giống nhau có thể được thực hiện bằng cách chỉ cần so sánh mỗi transaction với transaction tiếp theo trong cơ sở dữ liệu.

3.2.5 Cắt tỉa không gian tìm kiếm bằng Sub-tree Utility and Local Utility

Định nghĩa 21: Sub-tree utility

Cho itemset α và item z có thể mở rộng của α ($z \in E(\alpha)$). Sub-tree Utility của z đối với α được định nghĩa là:

$$su(\alpha, z) = \sum_{T \in g(\alpha \cup \{z\})} \left[u(\alpha, T) + u(z, T) + \sum_{i \in T \wedge i \in E(\alpha \cup \{z\})} u(i, T) \right] \quad (3.7)$$

trong đó $g(\alpha \cup \{z\})$ là tập hợp các transaction chứa $\alpha \cup \{z\}$.

Định lý 1: Giảm không gian tìm kiếm bằng sub-tree utility

Cho itemset α và item $z \in E(\alpha)$. Nếu $su(\alpha, z) < \delta$, thì itemset $\alpha \cup \{z\}$ và tất cả các itemset con của nó đều có giá trị lợi ích thấp hơn ngưỡng δ .

Dựa vào tính chất trên, có thể loại bỏ nhánh cây con của $\alpha \cup \{z\}$ khỏi không gian tìm kiếm.

Định nghĩa 22: Local utility.

Cho itemset α và item z có thể mở rộng của α ($z \in E(\alpha)$). Local Utility của z đối với α được định nghĩa là:

$$lu(\alpha, z) = \sum_{T \in g(\alpha \cup \{z\})} [u(\alpha, T) + re(\alpha, T)] \quad (3.8)$$

Định lý 2: Giảm không gian tìm kiếm bằng local utility

Cho itemset α và item $z \in E(\alpha)$. Nếu $lu(\alpha, z) < \delta$, thì tất cả các itemset con của α chứa z đều có giá trị lợi ích thấp hơn ngưỡng δ .

Dựa vào tính chất trên, có thể loại bỏ item z khỏi không gian tìm kiếm khi xét đến các itemset con của α .

3.2.6 Tính giới hạn trên hiệu quả bằng Fast Utility Counting

Fast Utility Counting (FUC), một phương pháp dựa trên mảng mới để tính toán hiệu quả các giới hạn trên (sub-tree utility và local utility) với độ phức tạp thời gian và không gian tuyến tính, sử dụng một cấu trúc dữ liệu mới gọi là utility-bin.

Định nghĩa 23: Utility-Bin

Cho tập các item I xuất hiện trong cơ sở dữ liệu D . Một mảng utility-bin U cho cơ sở dữ liệu D là một mảng có độ dài $|I|$, với mỗi phần tử được ký hiệu là $U[z]$ cho mỗi item $z \in I$. Mỗi phần tử được gọi là utility-bin và được sử dụng để lưu trữ một giá trị lợi ích.

Mảng utility-bin được sử dụng để lưu trữ và tính toán hiệu quả các giới hạn trên sau đây trong thời gian $O(|D|)$:

- Tính toán TWU của tất cả các item:
 1. Khởi tạo mảng utility-bin U .
 2. Với mỗi transaction T trong cơ sở dữ liệu, cập nhật utility-bin $U[z]$ cho mỗi item $z \in T$ như sau: $U[z] = U[z] + tu(T)$.
 3. Sau khi quét hết cơ sở dữ liệu, utility-bin $U[k]$ sẽ chứa $twu(k)$ cho mỗi mục $k \in I$.
- Tính toán sub-tree utility đối với một tập mục α :
 1. Khởi tạo mảng utility-bin U .
 2. Với mỗi transaction T trong cơ sở dữ liệu, cập nhật utility-bin $U[z]$ cho mỗi item $z \in T \cap E(\alpha)$ như sau: $U[z] = U[z] + u(\alpha, T) + u(z, T) + \sum_{i \in T \wedge i > z} u(i, T)$.
 3. Sau khi quét hết cơ sở dữ liệu, ta có $U[k] = su(\alpha, k)$, $\forall k \in I$.

- Tính toán local utility đối với một tập mục α :
 1. Khởi tạo mảng utility-bin U .
 2. Với mỗi transaction T trong cơ sở dữ liệu, cập nhật utility-bin $U[z]$ cho mỗi item $z \in T \cap E(\alpha)$ như sau: $U[z] = U[z] + u(\alpha, T) + re(\alpha, T)$.
 3. Sau khi quét hết cơ sở dữ liệu, ta có $U[k] = lu(\alpha, k)$, $\forall k \in I$.

3.2.7 Thuật toán EFIM

Phần này trình bày chi tiết về thuật toán EFIM [10], kết hợp tất cả các ý tưởng được giới thiệu trong các phần trước đó. Thuật toán nhận đầu vào là cơ sở dữ liệu transaction D và ngưỡng lợi ích tối thiểu δ do người dùng chỉ định. Ban đầu, thuật toán coi itemset hiện tại α là rỗng (dòng 1). Tiếp theo, thuật toán quét cơ sở dữ liệu một lần để tính toán local utility của mỗi item đối với α , sử dụng một mảng utility-bin (dòng 2). Trường hợp $\alpha = \emptyset$, local utility của một item chính là TWU của nó. Sau đó, local utility của mỗi item được so sánh với δ để xác định các secondary item đối với α , tức là các item cần được xem xét trong các phần mở rộng của α (dòng 3). Các secondary item sau đó được sắp xếp theo thứ tự tăng dần của TWU và thứ tự đó được sử dụng làm thứ tự \succ (dòng 4). Cơ sở dữ liệu sau đó được quét một lần để loại bỏ tất cả các item không phải là secondary item đối với α vì chúng không thể là một phần của bất kỳ itemset có giá trị lợi ích cao nào (theo định lý 2) (dòng 5). Nếu một transaction trở thành trống, nó sẽ bị xóa khỏi cơ sở dữ liệu. Sau đó, cơ sở dữ liệu được quét lại để sắp xếp các transaction theo thứ tự \succ_T để cho phép hợp nhất transaction trong $O(|D|)$ (dòng 6). Sau đó, thuật toán quét cơ sở dữ liệu lần nữa để tính toán sub-tree utility của mỗi secondary item đối với α , sử dụng mảng utility-bin (dòng 7). Các primary item đối với α sau đó được xác định (dòng 8). Cuối cùng, thuật toán gọi thuật toán đệ quy Search để thực hiện tìm kiếm theo chiều sâu bắt đầu từ α (dòng 9).

Mã giả của thuật toán EFIM được mô tả trong Thuật toán 1.

Thuật toán 1 EFIM

Input: D , the transaction database; δ , the minimum utility threshold.

Output: the set of high-utility itemsets.

- 1: $\alpha \leftarrow \emptyset$
 - 2: Calculate $lu(\alpha, i)$ for all items $i \in I$ by scanning D , using a utility-bin array
 - 3: $Secondary(\alpha) \leftarrow \{i \mid i \in I \wedge lu(\alpha, i) \geq \delta\}$
 - 4: Let \succ be the total order of TWU ascending values on $Secondary(\alpha)$
 - 5: Scan D to remove each item $i \notin Secondary(\alpha)$ from transactions
 - 6: Sort items in each transaction according to \succ and delete empty transactions
 - 7: Sort transactions in D according to \succ_T
 - 8: Calculate the sub-tree utility $su(\alpha, i)$ of each item $i \in Secondary(\alpha)$ by scanning D , using a utility-bin array
 - 9: $Primary(\alpha) \leftarrow \{i \mid i \in Secondary(\alpha) \wedge su(\alpha, i) \geq \delta\}$
 - 10: $Search(\alpha, D, Primary(\alpha), Secondary(\alpha), \delta)$
-

Thuật toán Search nhận các tham số là itemset hiện tại cần mở rộng α , cơ sở dữ liệu được chiếu theo α , các primary item và secondary item đối với α và ngưỡng lợi ích tối thiểu δ . Search lặp qua từng primary item i của α (dòng 1). Với mỗi primary item i , thuật toán tạo ra một itemset mở rộng β bằng cách thêm i vào α (dòng 2). Thuật toán quét cơ sở dữ liệu chiếu $\alpha - D$ để tính toán giá trị lợi ích của β và đồng thời xây dựng cơ sở dữ liệu được chiếu theo β ($\beta - D$), sử dụng kỹ thuật hợp nhất transaction (dòng 3). Nếu giá trị lợi ích của β không nhỏ hơn δ , β được xuất ra dưới dạng itemset có giá trị lợi ích cao (dòng 4). Thuật toán quét $\beta - D$ một lần để tính toán sub-tree utility và local utility của mỗi item z có thể mở rộng β (các secondary item đối với α), sử dụng hai mảng utility-bin (dòng 5). Dựa trên các giá trị sub-tree utility và local utility được tính toán, thuật toán xác định các primary item và secondary item của β (dòng 6-7). Cuối cùng, thuật toán Search được gọi đệ quy với β để tiếp tục tìm kiếm theo chiều sâu bằng cách mở rộng β (dòng 8). Mã giả của thuật toán Search được mô tả trong Thuật toán 2.

Thuật toán 2 Search

Input: α , an itemset; $\alpha - D$, the α projected database; $Primary(\alpha)$, the primary items of α ; $Secondary(\alpha)$, the secondary items of α ; δ , the minimum utility threshold.

Output: the set of high-utility itemsets that are extensions of α .

- 1: **for** each item $i \in Primary(\alpha)$ **do**
 - 2: $\beta \leftarrow \alpha \cup \{i\}$
 - 3: Scan $\alpha - D$ to calculate $u(\beta)$ and create $\beta - D$ \triangleright Uses transaction merging
 - 4: **if** $u(\beta) \geq \delta$ **then**
 - 5: Output β
 - 6: Calculate $su(\beta, z)$ and $lu(\beta, z)$ for all $z \in Secondary(\alpha)$ by scanning $\beta - D$ once, using two utility-bin arrays
 - 7: $Primary(\beta) \leftarrow \{z \in Secondary(\alpha) \mid su(\beta, z) \geq \delta\}$
 - 8: $Secondary(\beta) \leftarrow \{z \in Secondary(\alpha) \mid lu(\beta, z) \geq \delta\}$
 - 9: Search($\beta, \beta - D, Primary(\beta), Secondary(\beta), \delta$)
-

3.3 Bảo vệ tính riêng tư trong khai thác mẫu hữu ích

Trong thời đại số và bùng nổ dữ liệu, khai thác mẫu hữu ích giúp phát hiện các thông tin giá trị từ dữ liệu transaction, nhưng đồng thời cũng tạo ra nguy cơ rò rỉ thông tin nhạy cảm, đặc biệt trong môi trường thương mại. Để khắc phục điều này, các kỹ thuật bảo vệ tính riêng tư trong khai thác mẫu được phát triển nhằm ẩn thông tin nhạy cảm thông qua việc làm nhiều dữ liệu một cách có kiểm soát, đồng thời vẫn giữ được giá trị của các mẫu hữu ích.

Hai thuật toán PPUM được giới thiệu lần đầu tiên bởi Yeh và Hsu (2010), có tên là HHUIF và MSCIF [11]. Trong cả hai thuật toán, số lượng của các item mục tiêu được giảm cho đến khi giá trị lợi ích của các SHUI giảm xuống dưới ngưỡng lợi ích tối thiểu. Sự khác biệt giữa hai thuật toán nằm ở chiến lược chọn item mục tiêu: HHUIF chọn item có lợi ích cao nhất, trong khi MSICF chọn item xuất hiện nhiều nhất trong các itemset

nhạy cảm. Về kết quả, cả hai thuật toán đều có thời gian thực thi lớn trên cơ sở dữ liệu lớn do phải quét dữ liệu nhiều lần và gây mất mát một lượng lớn các itemset hữu ích không nhạy cảm.

Nhằm cải thiện hiệu suất của HHUIF, Yun và Kim (2015) đã đề xuất thuật toán FPUTT [12]. Thuật toán này sử dụng cấu trúc cây để giảm số lần quét cơ sở dữ liệu xuống còn ba lần, giúp tăng tốc đáng kể so với HHUIF. Tuy nhiên, do vẫn giữ nguyên chiến lược chọn item mục tiêu của HHUIF, FPUTT tiếp tục gặp phải các tác dụng phụ tương tự. Bên cạnh đó, cây FPUTT đòi hỏi không gian bộ nhớ lớn và chi phí tính toán cao.

Lin và cộng sự (2016) đã giới thiệu hai thuật toán mới là MSU-MAU và MSU-MIU [2] để giảm tác dụng phụ do ẩn các SHUI. Sự khác biệt giữa hai thuật toán này cũng nằm ở chiến lược chọn item mục tiêu: MSU-MAU chọn item có lợi ích lớn nhất trong transaction mục tiêu, trong khi MSU-MIU chọn item có lợi ích nhỏ nhất. Kết quả thực nghiệm cho thấy cả hai thuật toán không chỉ giảm tác dụng phụ tốt hơn HHUIF và MSICF mà còn cải thiện tốc độ nhờ sử dụng bảng chỉ mục để lưu trữ thông tin các transaction liên quan đến từng itemset nhạy cảm.

Ngoài các thuật toán dựa vào phương pháp heuristic ở trên, Li và cộng sự (2019) lần đầu tiên áp dụng quy hoạch số nguyên tuyến tính (Integer Linear Programming - ILP) để giải quyết vấn đề PPUM, với thuật toán là PPUM-ILP [13]. Ý tưởng là mô hình hóa quá trình ẩn các SHUI thành một bài toán thỏa mãn ràng buộc (Constraints Satisfaction Problem - CSP) và sử dụng cơ chế làm lỏng ràng buộc để tìm lời giải. Kết quả cho thấy, PPUM-ILP giúp giảm đáng kể các tác dụng phụ nhưng thời gian thực thi không ổn định, đặc biệt có thể kéo dài trên các tập dữ liệu lớn.

Để cải thiện hiệu suất của PPUM-ILP, Nguyen và cộng sự (2022) đã đề xuất thuật toán FILP [14], sử dụng cấu trúc dữ liệu băm để tăng tốc quá trình lọc các itemset và mô hình hóa bài toán. Nhờ đó, FILP đạt được hiệu suất cao hơn so với PPUM-ILP.

Tiếp nối, Nguyen và cộng sự (2023) đã phát triển thuật toán G-ILP [1]

nhằm giảm tác dụng phụ và tối ưu thời gian thực thi trong quá trình làm nhiễu. G-ILP áp dụng lập trình song song trên GPU để rút ngắn thời gian tiền xử lý và xây dựng một mô hình thỏa mãn ràng buộc khác hiệu quả hơn. Kết quả thực nghiệm chứng minh rằng G-ILP vượt trội cả về thời gian thực thi và khả năng giảm thiểu tác dụng phụ, đặc biệt trên các tập dữ liệu lớn và thừa, so với tất cả các thuật toán trước đó.

Tất cả các thuật toán được thảo luận ở trên đều áp dụng mô hình làm nhiễu dựa trên item để ẩn SHUI, trong đó số lượng của các item mục tiêu được sửa đổi. Ngược lại, một số nghiên cứu khác tuân theo mô hình nhiễu dựa trên transaction cho nhiệm vụ PPUM. Tiêu biểu trong số đó là hai thuật toán PPUMGA+insert [15] và PPUMGAT [3], cả hai đều dựa trên thuật toán di truyền (Genetic Algorithm - GA).

PPUMGA+insert [15] là một trong những thuật toán đầu tiên theo mô hình này, được Lin và cộng sự giới thiệu vào năm 2014. Thuật toán hoạt động bằng cách chèn thêm các transaction mới vào cơ sở dữ liệu nhằm tăng tổng lợi ích của cơ sở dữ liệu. Khi tổng lợi ích tăng, ngưỡng lợi ích tối thiểu cũng được nâng lên, từ đó làm cho các SHUI không còn được coi là lợi ích cao. Tuy nhiên, việc thêm các transaction mới có thể làm thay đổi cấu trúc dữ liệu và dẫn đến sự xuất hiện của các itemset hữu ích nhân tạo trong cơ sở dữ liệu sau khi làm nhiễu.

Tiếp nối, vào năm 2017, Lin và cộng sự đã phát triển một thuật toán khác có tên PPUMGAT [3]. Thuật toán này sử dụng cách tiếp cận đối lập so với PPUMGA+insert: thay vì thêm transaction, PPUMGAT thực hiện xóa các transaction hiện có trong cơ sở dữ liệu nhằm giảm lợi ích của SHUI. PPUMGAT sử dụng hàm mục tiêu để đánh giá mức độ phù hợp của các giải pháp ứng cử viên, từ đó xác định các transaction nhạy cảm cần xóa. Tuy nhiên, cũng giống như PPUMGA+insert, PPUMGAT phải đổi mới với vấn đề làm thay đổi số lượng transaction trong cơ sở dữ liệu, có thể dẫn đến mất mát thông tin và tạo ra các itemset hữu ích nhân tạo.

		HHUIF/ MSICF/ FPUTT	MSU- MAU	MSU- MIU	PPUM- ILP	FILP G-ILP	PPUMGA +insert	PPUMGAT
Mô hình nhiều Item	Tăng IU				X	X	X	
	Giảm IU	X	X	X	X	X	X	
	Transaction	Thêm				X		X
PP tối ưu	Xóa							
	Heuristics	X	X	X				
	Meta-heuristic						X	X
	CSP				X	X	X	

Bảng 3.1: Phân loại các thuật toán PPUM.

3.4 Một số thuật toán bảo vệ tính riêng tư trong khai thác mâu hữu ích

Phần này trình bày sơ lược về các thuật toán PPUM được sử dụng trong quá trình thực nghiệm của nghiên cứu.

3.4.1 MSU-MAU và MSU-MIU

MSU-MAU (Maximum Sensitive Utility - Maximum Item Utility) và MSU-MIU (Maximum Sensitive Utility - Minimum Item Utility) là hai thuật toán được đề xuất bởi Lin và cộng sự vào năm 2016 [2] nhằm giảm thiểu tác dụng phụ của quá trình ẩn dữ liệu nhạy cảm. Cả hai thuật toán đều dựa trên khái niệm MSU (Maximum Sensitive Utility), trong đó MSU của một SHUI là giá trị lợi ích lớn nhất của SHUI đó trong tất cả các transaction chứa nó. Bằng cách sử dụng MSU, các thuật toán này giúp ẩn các SHUI một cách hiệu quả mà không làm ảnh hưởng đáng kể đến cơ sở dữ liệu.

Quy trình thực hiện của MSU-MAU và MSU-MIU bao gồm hai bước chính. Đầu tiên, xét từng SHUI s và tính toán giá trị chênh lệch $diff$ giữa lợi ích của s và ngưỡng lợi ích tối thiểu δ , tức là $diff = u(s) - \delta$. Sau đó, thuật toán sẽ lặp lại cho đến khi $diff$ nhỏ hơn 0. Trong mỗi vòng lặp, transaction chứa s có giá trị MSU được xác định, gọi là transaction mục tiêu. Tùy thuộc vào thuật toán đang áp dụng, MSU-MAU sẽ chọn item có lợi ích lớn nhất trong transaction mục tiêu để chỉnh sửa, trong khi MSU-MIU sẽ chọn item có lợi ích nhỏ nhất. Việc chỉnh sửa item được thực hiện bằng cách xóa item khỏi transaction nếu lợi ích của nó nhỏ hơn $diff$, hoặc giảm internal utility của item đi một lượng $\left\lceil \frac{diff}{u(i)} \right\rceil$. Sau mỗi lần chỉnh sửa, giá trị $diff$ được cập nhật lại cho đến khi đạt yêu cầu.

Sự khác biệt chính giữa hai thuật toán nằm ở chiến lược lựa chọn item để chỉnh sửa. MSU-MAU ưu tiên chọn item có lợi ích lớn nhất trong

transaction mục tiêu nhằm nhanh chóng làm giảm lợi ích của itemset nhạy cảm s . Ngược lại, MSU-MIU chọn item có lợi ích nhỏ nhất để giảm thiểu tác động tiêu cực đến cơ sở dữ liệu. Việc chọn item có lợi ích nhỏ giúp đảm bảo rằng quá trình làm nhiễu ít ảnh hưởng đến các HUI khác trong cơ sở dữ liệu, đồng thời vẫn đạt được mục đích ẩn giấu SHUI.

Nhìn chung, cả hai thuật toán MSU-MAU và MSU-MIU đều đảm bảo ẩn giấu hoàn toàn các SHUI trong cơ sở dữ liệu đã làm nhiễu. So với các phương pháp trước đây như HHUIF và MSICF, hai thuật toán này giúp giảm thiểu tác động phụ đến cơ sở dữ liệu, duy trì chất lượng thông tin và đảm bảo tính hiệu quả của quá trình khai thác dữ liệu. Mã giả của hai thuật toán MSU-MAU và MSU-MIU được trình bày chi tiết trong Thuật toán 3 và Thuật toán 4.

Thuật toán 3 MSU-MAU

Input: D , the original database; I_H , the set of HUIs mined from D ; I_S , the set of SHUIs to be hidden; δ , the minimum utility threshold.

Output: D' , the sanitized database.

```

1: for each  $s \in I_S$  do
2:    $diff \leftarrow u(s) - \delta$ 
3:   while  $diff > 0$  do
4:      $T \mid i \leftarrow \text{argmax}_{s \subseteq T}(u(s, T))$ 
5:      $i \leftarrow \text{argmax}_{i \subseteq T \mid i}(u(i, T \mid i))$ 
6:     Modify  $o(i, T \mid i)$  with:
7:       
$$o(i, T \mid i) \leftarrow \begin{cases} 0, & \text{if } u(i, T \mid i) < diff \\ o(i, T \mid i) - \left\lceil \frac{diff}{u(i)} \right\rceil, & \text{if } u(i, T \mid i) > diff \end{cases}$$

8:       
$$diff \leftarrow \begin{cases} diff - u(i, T \mid i), & \text{if } u(i, T \mid i) < diff \\ 0, & \text{if } u(i, T \mid i) > diff \end{cases}$$

9: return the sanitized database  $D'$ 
```

Thuật toán 4 MSU-MIU

Input: D , the original database; I_H , the set of HUIs mined from D ; I_S , the set of SHUIs to be hidden; δ , the minimum utility threshold.

Output: D' , the sanitized database.

```
1: for each  $s \in I_S$  do
2:    $diff \leftarrow u(s) - \delta$ 
3:   while  $diff > 0$  do
4:      $T \mid i \leftarrow \text{argmax}_{s \subseteq T}(u(s, T))$ 
5:      $i \leftarrow \text{argmin}_{i \subseteq T \mid i}(u(i, T \mid i))$ 
6:     Modify  $o(i, T \mid i)$  with:
7:       
$$o(i, T \mid i) \leftarrow \begin{cases} 0, & \text{if } u(i, T \mid i) < diff \\ o(i, T \mid i) - \left\lceil \frac{diff}{u(i)} \right\rceil, & \text{if } u(i, T \mid i) > diff \end{cases}$$

8:      $diff \leftarrow \begin{cases} diff - u(i, T \mid i), & \text{if } u(i, T \mid i) < diff \\ 0, & \text{if } u(i, T \mid i) > diff \end{cases}$ 
9: return the sanitized database  $D'$ 
```

3.4.2 FILP

FILP [14] là một thuật toán PPUM dựa trên quy hoạch số nguyên tuyến tính (Integer Linear Programming - ILP), được giới thiệu năm 2022 bởi Nguyen và Le. FILP sử dụng cấu trúc bảng băm IT (Itemset-Tidset Table) để cải thiện hiệu năng so với thuật toán PPUM-ILP dựa trên ILP trước đó. Thuật toán này tập trung vào việc ẩn các itemset nhạy cảm trong khi vẫn đảm bảo tính chính xác của các itemset không nhạy cảm, giúp duy trì các mẫu quan trọng mà không gây ảnh hưởng không cần thiết.

FILP hoạt động thông qua bốn bước chính, trong đó bước đầu tiên là xây dựng bảng IT. Bảng IT được sử dụng để lưu trữ thông tin về các itemset và các transaction hỗ trợ chúng, với hai loại bảng chính: bảng S-IT (Sensitive IT) dành cho SHUI và bảng N-IT (Non-sensitive IT) dành cho NSHUI. Mỗi bảng bao gồm ba cột chính: itemset được băm, tập hợp ID của các transaction hỗ trợ itemset đó, và giá trị lợi ích của itemset. Việc sử dụng cấu trúc băm giúp FILP tăng tốc độ truy vấn và xử lý dữ liệu. Quá trình xây dựng bảng IT được thực hiện thông qua một lần quét cơ sở

dữ liệu gốc để trích xuất thông tin cần thiết, từ đó xây dựng bảng S-IT và N-IT một cách hiệu quả. Chi tiết quá trình xây dựng bảng IT được trình bày trong Thuật toán 5.

Thuật toán 5 Table Construction

Input: D , the original database; I_H , the set of HUIs mined from D ; I_S , the set of SHUIs to be hidden.

Output: N-IT, S-IT tables.

```

1: for each transaction  $T_n \in D$  do
2:   for each itemset  $X \in I_H$  do
3:     if  $X \subseteq T_n$  then
4:       if  $X \in I_S$  then
5:         Insert  $X$ , TID of  $T_n$ ,  $U(X)$  into S-IT.
6:       else
7:         Insert  $X$ , TID of  $T_n$ ,  $U(X)$  into N-IT.
8:   Hash stored tidsets and insert them into IT tables

```

Sau khi bảng IT được thiết lập, FILP thực hiện bước tiền xử lý nhằm mục đích giảm kích thước bài toán và loại bỏ các ràng buộc không cần thiết, giúp quá trình ẩn SHUI hiệu quả hơn. Trong bước này, thuật toán áp dụng ba chiến lược sau để lọc các NSHUI:

1. Lọc bỏ NSHUI không bị ảnh hưởng bởi quá trình ẩn SHUI, giúp giảm số lượng ràng buộc cần xử lý. Một NSHUI n được coi là bị ảnh hưởng bởi quá trình ẩn nếu nó chia sẻ ít nhất một item và một transaction với bất kỳ SHUI s nào.
2. Xác định và loại bỏ các itemset "bound-to-lose", giúp tránh xung đột khi thiết lập ràng buộc. Một itemset X được gọi là "bound-to-lose" nếu thỏa mãn đồng thời các điều kiện sau: X là một HUI, X là tập con của một itemset nhạy cảm Y , và tập hợp các transaction hỗ trợ X trùng khớp với tập hỗ trợ của Y (tức là $T_X = T_Y$).
3. Loại bỏ các itemset dư thừa, nhằm giảm số lượng ràng buộc không cần thiết. Nếu hai NSHUI n_{i1} và n_{i2} chia sẻ cùng một tập transaction $T_{n_{i1}}$, và n_{i1} là tập con của n_{i2} , thì n_{i2} sẽ bị loại bỏ.

Nhờ các bước tiền xử lý này, FILP có thể giảm đáng kể kích thước của bảng N-IT, qua đó cắt giảm số lượng ràng buộc trong bài toán CSP, loại bỏ các ràng buộc dư thừa và cải thiện hiệu suất giải quyết bài toán. Mã giả mô tả quá trình tiền xử lý được trình bày trong Thuật toán 6.

Thuật toán 6 Preprocessing

Input: N-IT, S-IT tables.

Output: Modified N-IT, S-IT tables.

```

1: for each itemset  $n_i$  in N-IT do
2:    $L \leftarrow \emptyset$ 
3:   for each itemset  $s_i$  in S-IT do
4:     if  $n_i \cap s_i \neq \emptyset$  and  $T_{n_i} \cap T_{s_i} \neq \emptyset$  then
5:       if  $n_i \subseteq s_i$  and  $T_{n_i} = T_{s_i}$  then
6:         Delete  $n_i$  from N-IT
7:        $L \leftarrow L + 1$ 
8:     if  $L = I_S$  then
9:       Delete  $n_i$  from N-IT
10:    for each itemset  $n_{i_2}$  in N-IT do
11:      for each itemset  $n_{i_1}$  in N-IT do
12:        if  $n_{i_1} \neq n_{i_2}$  and  $n_{i_1} \subseteq n_{i_2}$  and  $T_{n_{i_1}} = T_{n_{i_2}}$  then
13:          Delete  $n_{i_2}$  from N-IT
14:          break
```

Bước quan trọng tiếp theo trong thuật toán FILP là xây dựng bài toán dưới dạng mô hình thỏa mãn ràng buộc, có thể được giải quyết bằng kỹ thuật lập trình nguyên. Mục đích của bước này là tìm ra giải pháp tối ưu để ẩn SHUI trong khi vẫn giảm thiểu tác động tiêu cực đến NSHUI. Quá trình xây dựng bài toán bao gồm ba thành phần chính: thay thế giá trị lợi ích nội của SHUI bằng các biến nguyên, xây dựng ràng buộc và thiết lập hàm mục tiêu.

Trước tiên, FILP thay thế các giá trị lợi ích nội của SHUI bằng các biến nguyên. Cụ thể, đối với mỗi SHUI trong bảng S-IT, lợi ích nội của các item thuộc SHUI trong từng transaction sẽ được thay thế bằng một biến nguyên, đại diện cho giá trị cần điều chỉnh. Tập hợp các biến này

được ký hiệu là $V = \{v_{nm} \mid n \in [1, N] \cap \mathbb{N}, m \in [1, M] \cap \mathbb{N}\}$, trong đó v_{nm} thể hiện lợi ích nội của item m trong transaction n . Nhằm đảm bảo SHUI sau điều chỉnh không bị xóa khỏi các transaction, giá trị tối thiểu của mỗi biến nguyên được đặt là 1. Để tăng tốc quá trình truy cập và xử lý, thuật toán sử dụng một bảng băm lưu trữ vị trí và giá trị của các biến nguyên.

Sau khi thiết lập các biến nguyên, FILP xây dựng hệ thống ràng buộc để đảm bảo việc ẩn SHUI và duy trì lợi ích của NSHUI. Để ẩn SHUI X , tổng lợi ích của X trong tất cả các transaction phải nhỏ hơn ngưỡng lợi ích tối thiểu δ . Ràng buộc này được biểu diễn bởi công thức:

$$\sum_{T_n \in D, X \subseteq T_n} \sum_{i_m \in X} v_{nmp}(i_m) < \delta \quad (3.9)$$

Trong khi đó, để đảm bảo tính toàn vẹn của NSHUI Y , tổng lợi ích của Y sau khi điều chỉnh phải lớn hơn hoặc bằng ngưỡng tối thiểu δ . Điều kiện này được thể hiện bởi công thức:

$$\sum_{T_{n'} \in D, Y \subseteq T_{n'}} \sum_{i_{m'} \in Y} u_{n'm'}(i_{m'}) \geq \delta \quad (3.10)$$

Trong đó, giá trị $u_{n'm'}$ được xác định như sau: nếu $v_{n'm'} \in V$, thì $u_{n'm'} = v_{n'm'}$, ngược lại $u_{n'm'} = q(i_{m'}, T_{n'})$.

Thuật toán 7 CSP Formulation

Input: S-IT, N-IT tables

Output: CSP model

- 1: Using the S-IT table create a hash table V to store variable positions and utilities; establish inequalities according to formula 3.9
- 2: **for** each itemset n_i in N-IT **do**
- 3: Subtract the sum of the variable utilities in n_i from n_i utility
- 4: Establish inequalities according to formula 3.10
- 5: **end**

Bên cạnh các ràng buộc, FILP còn thiết lập một hàm mục tiêu nhằm giảm thiểu tổng các biến nguyên, tương đương với việc tối ưu hóa lượng điều chỉnh lợi ích nội. Điều này giúp giảm thiểu Artificial Cost, tức là lượng thông tin dư thừa được thêm vào cơ sở dữ liệu sau khi điều chỉnh. Hàm mục tiêu được định nghĩa như sau:

$$\min \sum_{v_{nm} \in v} v_{nm} \quad (3.11)$$

Bước cuối cùng trong thuật toán FILP là giải bài toán thỏa mãn ràng buộc (CSP) đã được xây dựng ở bước trước. Mục tiêu của bước này là tìm ra một bộ giá trị cho các biến nguyên, đảm bảo thỏa mãn tất cả các ràng buộc đồng thời tối ưu hóa hàm mục tiêu. FILP sử dụng Gurobi 9.1.2, một công cụ giải lập trình nguyên hiệu năng cao, để xử lý bài toán. Nếu bài toán có giải pháp khả thi, Gurobi sẽ tìm ra một giải pháp tối ưu. Tuy nhiên, nếu bài toán không khả thi, Gurobi sẽ đưa ra thông báo, khi đó FILP áp dụng phương pháp nổi lỏng ràng buộc nhằm tìm kiếm một giải pháp gần đúng. Phương pháp này được thực hiện bằng cách loại bỏ dần các ràng buộc trong bài toán CSP, sau đó lặp lại quá trình giải bằng Gurobi cho đến khi tìm thấy một giải pháp phù hợp. Sau khi có được giải pháp thích hợp, FILP sử dụng các giá trị của biến nguyên trong giải pháp để điều chỉnh lợi ích nội của các item thuộc SHUI trong cơ sở dữ liệu gốc, từ đó tạo ra phiên bản cơ sở dữ liệu đã được điều chỉnh.

Tóm lại, FILP là một thuật toán PPUM hiệu quả, sử dụng cấu trúc dữ liệu thông minh và kỹ thuật lập trình nguyên để ẩn giấu thông tin nhạy cảm. Kết quả thực nghiệm cho thấy FILP có hiệu suất vượt trội so với thuật toán PPUM-ILP. Mặc dù vậy, FILP vẫn tồn tại một số hạn chế cần được khắc phục trong tương lai để nâng cao hơn nữa hiệu quả và khả năng áp dụng vào thực tế.

3.4.3 PPUMGAT

PPUMGAT [3] (Privacy-Preserving Utility Mining using Genetic Algorithm with Transaction Deletion) là một thuật toán ẩn giấu các itemset có giá trị lợi ích cao nhạy cảm thông qua việc xóa transaction, được phát triển bởi Lin và cộng sự vào năm 2017. Thuật toán này ứng dụng thuật toán di truyền kết hợp với khái niệm pre-large để tối ưu hóa quá trình tìm kiếm giải pháp.

Quá trình hoạt động của PPUMGAT bắt đầu bằng việc xác định MDU (Maximum Deleted Utility) – tổng lợi ích tối đa cần xóa để đảm bảo rằng các SHUI không còn vượt ngưỡng lợi ích tối thiểu δ . MDU được tính toán dựa trên sự chênh lệch giữa lợi ích có trọng số transaction (TWU) của các itemset nhạy cảm và ngưỡng δ , theo công thức:

$$MDU = \sum_{s \in I_S} (TWU(s) - TU \times \delta) \quad (3.12)$$

Giá trị MDU đóng vai trò như một "ngưỡng an toàn", giúp thuật toán xác định lượng lợi ích cần loại bỏ để ẩn tất cả các SHUI mà không làm suy giảm quá mức giá trị tổng thể của cơ sở dữ liệu.

Sau khi tính toán MDU, thuật toán xây dựng tập hợp Candi_Delete - tập các transaction ứng viên để xóa. Một transaction được đưa vào Candi_Delete nếu nó chứa ít nhất một SHUI và có lợi ích transaction (tu) thấp hơn MDU. Điều này giúp đảm bảo việc xóa transaction không chỉ tập trung vào thông tin nhạy cảm mà còn hạn chế ảnh hưởng đến các itemset quan trọng khác trong cơ sở dữ liệu.

Trọng tâm của PPUMGAT là ứng dụng thuật toán di truyền để tìm ra tập transaction tối ưu cần xóa. Mỗi nhiễm sắc thể trong GA biểu diễn một tập hợp các transaction được đề xuất loại bỏ. Độ dài của nhiễm sắc thể được xác định bằng cách sắp xếp các transaction trong Candid_Delete theo thứ tự tăng dần của giá trị tu , sau đó cộng dồn giá trị tu cho đến khi tổng lớn hơn MDU. Số lượng transaction đã cộng dồn sẽ được sử dụng làm

độ dài nhiễm sắc thể. Quần thể ban đầu được khởi tạo ngẫu nhiên từ tập Candi_Delete, và chất lượng của mỗi nhiễm sắc thể được đánh giá thông qua hàm fitness.

Hàm fitness cân bằng ba yếu tố: (1) số lượng HUI nhạy cảm chưa được ẩn (α), (2) số lượng HUI không nhạy cảm bị mất (β), và (3) số lượng HUI giả xuất hiện sau quá trình xóa (γ). Các trọng số w_1, w_2, w_3 trong hàm fitness cho phép người dùng điều chỉnh mức độ ưu tiên giữa các yếu tố này tùy theo yêu cầu cụ thể của bài toán. Hàm fitness được biểu diễn như sau:

$$\text{fitness} = w_1 \times \alpha + w_2 \times \beta + w_3 \times \gamma \quad (3.13)$$

Trong quá trình tiến hóa, thuật toán thực hiện các phép toán di truyền như lai ghép (crossover) và đột biến (mutation) nhằm tạo ra các thế hệ mới với tiềm năng cải thiện giải pháp. Lai ghép giúp kết hợp các nhiễm sắc thể cha mẹ để tạo ra thế hệ con có đặc điểm tối ưu hơn, trong khi đột biến thay đổi ngẫu nhiên một số gen nhằm đảm bảo sự đa dạng trong quần thể, tránh rơi vào các cực trị cục bộ. Sau mỗi vòng tiến hóa, thuật toán chọn ra một nửa số nhiễm sắc thể có giá trị fitness tốt nhất và tạo mới một nửa còn lại để duy trì sự đa dạng, đảm bảo sự cải thiện trong quá trình tìm kiếm lời giải. Quá trình tiến hóa tiếp tục cho đến khi đạt điều kiện dừng, chẳng hạn như khi đạt đến số vòng lặp tối đa hoặc khi không còn cải thiện đáng kể về giá trị fitness.

Một cải tiến quan trọng của thuật toán PPUMGAT là việc sử dụng khái niệm pre-large để tối ưu hóa hiệu suất. Khái niệm này sử dụng hai ngưỡng lợi ích (S_u và S_l) để duy trì một bộ đệm chứa các itemset có "tiềm năng" trở thành HUI nhân tạo sau khi xóa transaction. Các itemset này được gọi là pre-large itemsets (PUIs), có lợi ích nằm trong khoảng $[S_l, S_u]$. Bằng cách sử dụng pre-large, thuật toán có thể tính toán tác dụng phụ một cách hiệu quả hơn, tránh việc quét lại toàn bộ cơ sở dữ liệu trong mỗi vòng lặp, giúp tiết kiệm đáng kể thời gian và tài nguyên tính toán, đặc biệt là với các cơ sở dữ liệu lớn.

Sở dĩ PPUMGAT có thể làm xuất hiện các HUI nhân tạo là vì nó sử dụng ngưỡng lợi ích tối thiểu $\delta = c \times TU$ (c là hằng số). Khi thuật toán thực hiện xóa transaction, tổng lợi ích của cơ sở dữ liệu TU sẽ bị giảm, dẫn đến ngưỡng lợi ích tối thiểu δ cũng giảm theo. Chính điều này là nguyên nhân làm xuất hiện các HUI nhân tạo. Để đồng bộ với các thuật toán được sử dụng cho quá trình thực nghiệm ở trên, PPUMGAT sẽ được điều chỉnh để sử dụng ngưỡng lợi ích tối thiểu δ là một hằng số.

Khi quá trình tiến hóa kết thúc, nhiệm sắc thẻ tối ưu với giá trị fitness thấp nhất sẽ được chọn làm giải pháp cuối cùng. Các transaction trong nhiệm sắc thẻ này sẽ bị xóa khỏi cơ sở dữ liệu để đảm bảo rằng các itemset nhạy cảm đã được ẩn thành công.

Tóm lại, PPUMGAT là một phương pháp đột phá trong lĩnh vực PPUM. Bằng cách kết hợp giữa thuật toán di truyền, kỹ thuật xóa transaction có chọn lọc, và khái niệm pre-large, thuật toán này không chỉ bảo vệ thông tin nhạy cảm mà còn duy trì giá trị hữu ích của dữ liệu. Mã giả của thuật toán PPUMGAT được trình bày chi tiết trong Thuật toán 8.

3.5 Tối ưu hóa ngẫu nhiên

Tối ưu hóa ngẫu nhiên (Stochastic Optimization) là một nhánh quan trọng của tối ưu hóa, với mục tiêu tìm ra giải pháp tối ưu hoặc gần tối ưu cho các bài toán mà không gian giải pháp quá lớn, quá phức tạp hoặc chứa nhiều cực trị cục bộ. Đặc điểm nổi bật của các thuật toán tối ưu hóa ngẫu nhiên là khả năng thêm các yếu tố ngẫu nhiên vào quá trình tìm kiếm, giúp các thuật toán có thể thoát khỏi các bẫy cực trị cục bộ và tiến gần hơn tới cực trị toàn cục. Nhờ đó, các thuật toán này không chỉ hiệu quả trong các bài toán tối ưu thông thường mà còn đặc biệt hữu ích với những bài toán phi tuyến hoặc không lồi, nơi các phương pháp tối ưu hóa cổ điển thường gặp khó khăn.

Hầu hết các thuật toán tối ưu hóa ngẫu nhiên lấy cảm hứng từ tự nhiên

Thuật toán 8 PPUMGAT

Input: D , the original database; I_H , the set of HUIs mined from D ; I_S , the set of SHUIs to be hidden; δ , the minimum utility threshold; M , number of chromosomes; N , number of iterations.

Output: D' , the sanitized database.

```
1: for each  $s_i \in I_S$  do
2:    $MDU \leftarrow TWU(s_i) - TU \times \delta$ 
3:   for each transaction  $T_q \in D$  do
4:     Calculate  $tu(T_q)$ 
5:     for each  $s_i \in I_S$  do
6:       if  $s_i \in T_q$  and  $tu(T_q) < MDU$  then
7:         Add  $T_q$  to Candi_Delete
8: Sort transactions in Candi_Delete in ascending order of  $tu$ 
9: Set  $m = 0$ , sum = 0
10: for each  $T_q$  in Candi_Delete do
11:   if sum < MDU then
12:     sum +=  $tu(T_q)$ 
13:      $m = m + 1$ 
14: Set chromosome size to  $m$ 
15: Randomly generate  $M$  chromosomes as initial population
16: while termination criteria not met do
17:   for each chromosome  $c_i$  in population do
18:     Perform crossover operation
19:     Perform mutation operation
20:     Evaluate fitness( $c_i$ )
21:   Select top  $M/2$  chromosomes
22:   Generate  $M/2$  new chromosomes for next generation
23:   Obtain optimal chromosome  $c^*$  with minimal fitness
24:   Delete transactions of  $c^*$  from  $D$  to obtain  $D'$ 
25: return  $D'$ 
```

và thế giới thực. Chúng mô phỏng các hiện tượng sinh học, vật lý hoặc hành vi xã hội để xây dựng các cơ chế tìm kiếm lời giải. Các thuật toán này bắt đầu bằng việc khởi tạo ngẫu nhiên một tập hợp các giải pháp ban đầu. Qua nhiều vòng lặp, các giải pháp được cải thiện thông qua các chiến lược đặc thù như chọn lọc tự nhiên, trao đổi thông tin, hoặc sử dụng các phép toán xác suất. Yếu tố ngẫu nhiên được từng bước tích hợp để mở rộng phạm vi tìm kiếm và tăng đa dạng trong các giải pháp, giúp tăng khả năng tìm ra giải pháp tối ưu toàn cục.

Các thuật toán tối ưu hóa ngẫu nhiên đã phát triển qua nhiều thế hệ, với nhiều cơ chế sáng tạo tạo và đa dạng. Một trong những thuật toán tiêu biểu là Genetic Algorithm (GA) [16], được đề xuất bởi Holland vào năm 1975. GA dựa trên lý thuyết chọn lọc tự nhiên và di truyền học, mô phỏng quá trình tiến hóa của quần thể. Trong đó, các cá thể mạnh hơn có cơ hội sinh sản cao hơn, từ đó chất lượng của các giải pháp được cải thiện qua từng thế hệ. Một phương pháp khác, Simulated Annealing (SA) [17], được Kirkpatrick, Gelatt, và Vecchi giới thiệu vào năm 1983, lại mô phỏng quá trình làm nguội trong luyện kim. SA sử dụng cơ chế giảm dần "nhiệt độ" để mở rộng không gian tìm kiếm, giúp tránh được các bẫy cực trị cục bộ.

Ant Colony Optimization (ACO) [18], được Dorigo giới thiệu vào năm 1992, bắt chước cách loài kiến tìm kiếm thức ăn thông qua việc để lại pheromone trên đường đi. Cơ chế tích lũy và bốc hơi pheromone giúp ACO tìm ra các lộ trình tối ưu trong không gian tìm kiếm một cách hiệu quả. Một thuật toán nổi bật khác là Particle Swarm Optimization (PSO) [19], được phát triển bởi Kennedy và Eberhart vào năm 1995, lấy cảm hứng từ hành vi bầy đàn của các loài chim hoặc cá. PSO sử dụng một tập hợp các "hạt" di chuyển trong không gian tìm kiếm, nơi mỗi hạt điều chỉnh vị trí của mình dựa trên kinh nghiệm cá nhân và tập thể. Bên cạnh đó, Harmony Search (HS) [20], được phát triển bởi Geem, Kim, và Loganathan vào năm 2001, lấy cảm hứng từ cách các nhạc công điều chỉnh giai điệu để đạt được sự hài hòa. HS khai thác không gian giải pháp theo cách ngẫu

nhiên nhưng có định hướng rõ ràng, tạo ra sự cân bằng giữa tìm kiếm toàn cục và khai thác cục bộ.

Ngoài các thuật toán kinh điển, những phương pháp hiện đại đã được phát triển để giải quyết các bài toán tối ưu hóa phức tạp hơn. Grey Wolf Optimizer (GWO) [21], được Mirjalili giới thiệu vào năm 2014, mô phỏng cấu trúc lãnh đạo của bầy sói xám. GWO dựa vào hành vi săn mồi và tổ chức theo thứ bậc trong bầy để hướng dẫn tìm kiếm giải pháp. Tiếp theo, Ant Lion Optimizer (ALO) [22], cũng được Mirjalili phát triển vào năm 2015, lấy cảm hứng từ hành vi săn mồi của kiến sư tử, cân bằng giữa khám phá không gian tìm kiếm và khai thác các vùng tiềm năng. Cuối cùng, Spotted Hyena Optimizer (SHO) [23], do Dhiman và Kumar đề xuất vào năm 2017, tái hiện hành vi săn mồi và hợp tác của linh cẩu đốm, tận dụng tốt sự tương tác giữa các thành viên trong bầy để cải thiện chất lượng giải pháp.

Ứng dụng của các thuật toán tối ưu hóa ngẫu nhiên trải rộng trong nhiều lĩnh vực khác nhau. Trong học máy, các thuật toán như PSO và GA thường được sử dụng để tối ưu hóa siêu tham số của các mô hình hoặc lựa chọn đặc trưng. Trong kỹ thuật, tối ưu hóa ngẫu nhiên hỗ trợ thiết kế cấu trúc sản phẩm, nâng cao độ bền và hiệu suất, đồng thời tối ưu hóa chi phí sản xuất. Trong quản lý chuỗi cung ứng, ACO đã được ứng dụng để giải quyết bài toán tối ưu hóa lộ trình vận chuyển, góp phần giảm thiểu chi phí và thời gian giao hàng. Trong lĩnh vực tài chính, các thuật toán tối ưu hóa ngẫu nhiên được sử dụng để lập danh mục đầu tư, dự đoán xu hướng thị trường, và quản lý rủi ro. Ngoài ra, tối ưu hóa ngẫu nhiên cũng được áp dụng rộng rãi trong bảo vệ quyền riêng tư khi khai thác dữ liệu, giúp đảm bảo an toàn thông tin mà không làm mất đi giá trị của dữ liệu.

Tuy nhiên, không thể phủ nhận rằng các thuật toán tối ưu hóa ngẫu nhiên vẫn tồn tại một số hạn chế. Do tính ngẫu nhiên, chúng không đảm bảo luôn tìm được giải pháp tối ưu toàn cục trong thời gian hữu hạn. Hơn nữa, việc điều chỉnh các tham số như tỷ lệ đột biến, tốc độ hội tụ, hay số vòng lặp tối ưu hóa có thể trở nên phức tạp, đặc biệt khi áp dụng trên các

bài toán lớn. Dẫu vậy, nhờ khả năng kết hợp tính ngẫu nhiên với các chiến lược mô phỏng tự nhiên, tối ưu hóa ngẫu nhiên vẫn là một công cụ mạnh mẽ và có tiềm năng lớn trong việc giải quyết những thách thức trong khoa học và công nghệ.

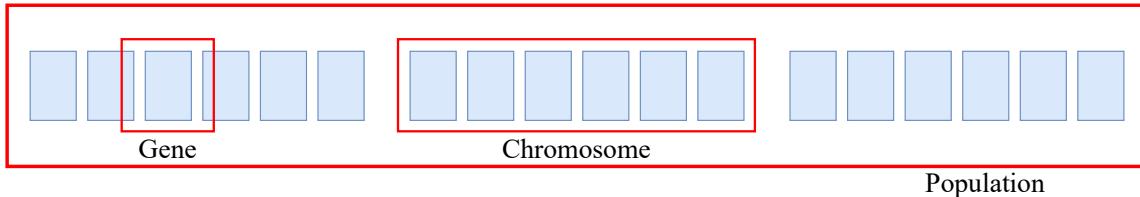
3.6 Một số thuật toán tối ưu hóa ngẫu nhiên

3.6.1 Genetic Algorithm

Genetic Algorithm [16] (thuật toán di truyền) là một phương pháp tối ưu hóa mạnh mẽ, lấy cảm hứng từ quy trình tiến hóa tự nhiên. Ý tưởng cốt lõi của GA là mô phỏng sự chọn lọc tự nhiên, trong đó những cá thể mạnh hơn có khả năng tồn tại và sinh sản cao hơn, dẫn đến sự tiến hóa của quần thể qua các thế hệ. Trong thuật toán này, mỗi giải pháp ứng viên được xem như một cá thể, và tập hợp các cá thể tạo thành một quần thể. Mỗi cá thể mang trong mình một bộ nhiễm sắc thể có thể được biểu diễn dưới dạng chuỗi nhị phân hoặc các dạng mã hóa khác, tùy thuộc vào đặc điểm của bài toán.

Quá trình tiến hóa trong GA bắt đầu từ một quần thể các cá thể được tạo ngẫu nhiên và diễn ra theo chu kỳ qua các thế hệ. Trong mỗi thế hệ, độ thích nghi của từng cá thể được đánh giá thông qua một hàm thích nghi, thường là giá trị của hàm mục tiêu trong bài toán tối ưu hóa. Các cá thể có độ thích nghi cao hơn có cơ hội lớn hơn được chọn để tạo ra thế hệ tiếp theo thông qua các toán tử di truyền như lai ghép và đột biến.

Điểm quan trọng trong GA là cách biểu diễn di truyền, tức là cách mã hóa giải pháp thành bộ nhiễm sắc thể, hay còn gọi là chuỗi gen, để thuật toán có thể xử lý. Phương pháp phổ biến nhất là sử dụng mảng các bit (0 và 1), được ưa chuộng nhờ tính đơn giản và hiệu quả trong việc thực hiện các toán tử di truyền. Tuy nhiên, tùy thuộc vào đặc điểm của bài toán, các dạng biểu diễn khác như mảng số thực, cấu trúc cây, hoặc đồ thị cũng có thể được áp dụng.

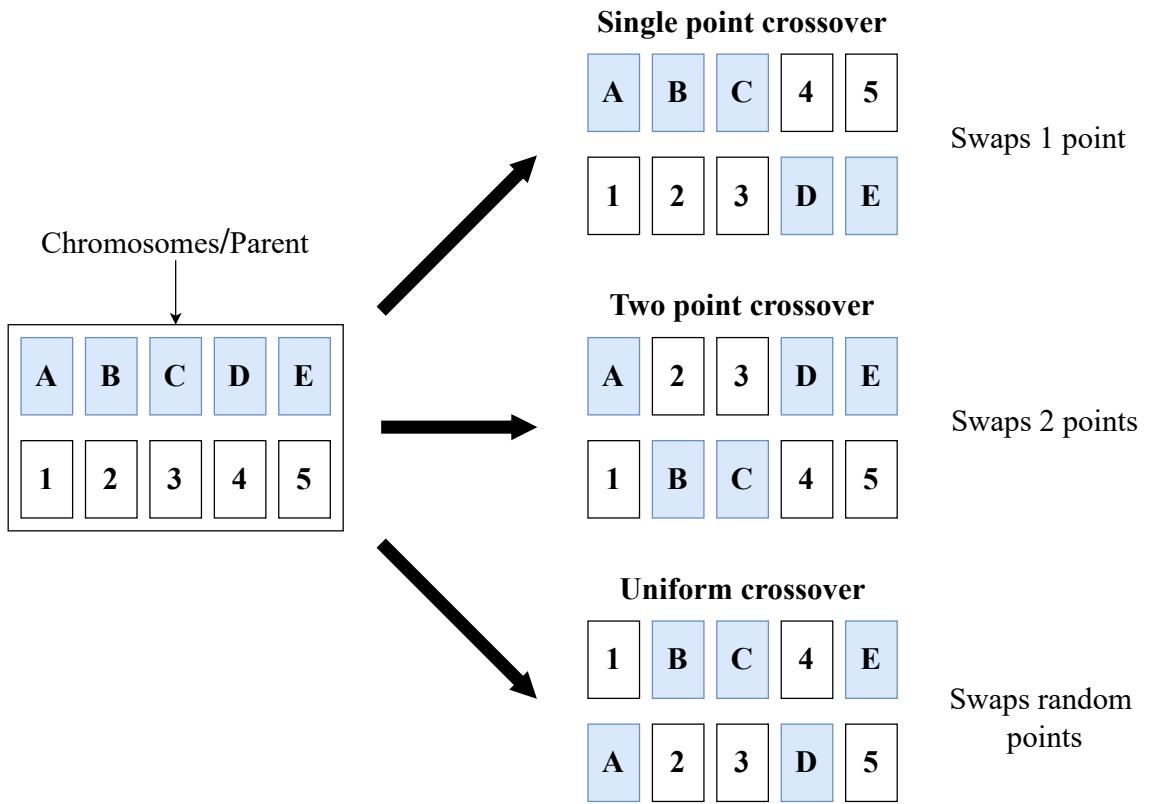


Hình 3.2: Một ví dụ về biểu diễn di truyền trong GA.

Sau khi xác định được biểu diễn di truyền, GA tiến hành khởi tạo quần thể ban đầu. Kích thước quần thể phụ thuộc vào bản chất của bài toán, thường dao động từ vài trăm đến vài nghìn cá thể. Thông thường, quần thể ban đầu được tạo ra một cách ngẫu nhiên nhằm đảm bảo độ đa dạng và bao phủ toàn bộ không gian tìm kiếm. Tuy nhiên, trong một số trường hợp đặc biệt, các giải pháp có thể được "gieo" trong những vùng có khả năng chứa giải pháp tối ưu, giúp tăng tốc độ hội tụ của thuật toán.

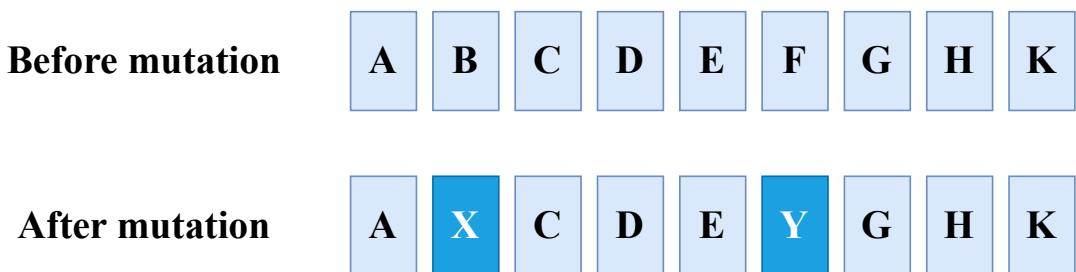
Chọn lọc là quá trình quan trọng trong việc xác định những cá thể nào sẽ được chọn để tạo ra thế hệ tiếp theo. Quá trình này dựa trên độ thích nghi của từng cá thể, được đánh giá thông qua hàm thích nghi. Các cá thể có độ thích nghi cao hơn sẽ có xác suất lớn hơn được chọn để sinh sản. Một số phương pháp chọn lọc đánh giá toàn bộ quần thể và ưu tiên lựa chọn các cá thể tốt nhất, trong khi các phương pháp khác chỉ tập trung vào một mẫu ngẫu nhiên của quần thể nhằm giảm thời gian tính toán. Một số phương pháp chọn lọc phổ biến bao gồm phương pháp vòng quay bánh xe, phương pháp giải đấu, và phương pháp chọn lọc dựa trên thứ hạng.

Lai ghép là quá trình kết hợp thông tin di truyền từ hai hoặc nhiều cá thể cha mẹ để tạo ra cá thể con mới. Các cá thể con thường thừa hưởng nhiều đặc điểm di truyền từ cha mẹ, giúp chúng có tiềm năng trở thành giải pháp tốt hơn. Có nhiều phương pháp lai ghép như lai ghép một điểm, lai ghép nhiều điểm, và lai ghép đồng nhất. Mỗi phương pháp có những ưu điểm riêng và phù hợp với những loại bài toán khác nhau. Hình 3.3 dưới đây minh họa một số phương pháp lai ghép phổ biến.



Hình 3.3: Một số phương pháp lai ghép phổ biến trong GA.

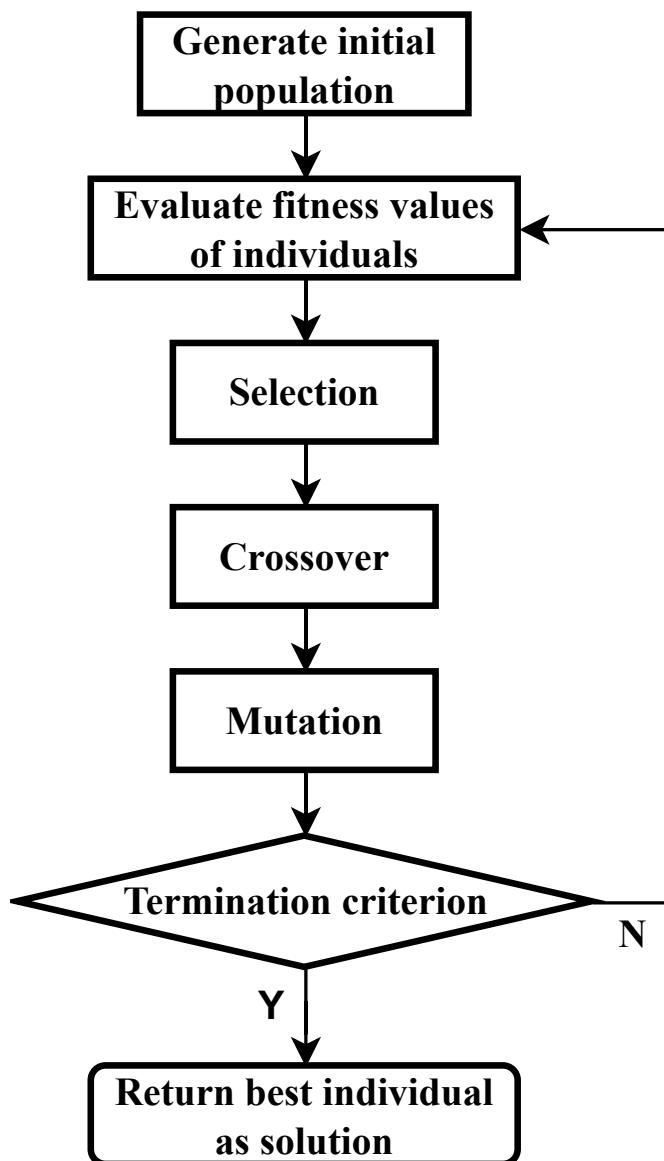
Đột biến là quá trình tạo ra những thay đổi ngẫu nhiên trong nhiễm sắc thể của các cá thể. Quá trình này đóng vai trò quan trọng trong việc duy trì sự đa dạng di truyền của quần thể, giúp thuật toán thoát khỏi các cực trị địa phương. Tỷ lệ đột biến cần được điều chỉnh cẩn thận: tỷ lệ quá thấp có thể dẫn đến hiện tượng trôi dạt di truyền, trong khi tỷ lệ quá cao có thể làm mất đi những giải pháp tốt đã tìm được. Hình 3.4 dưới đây minh họa quá trình đột biến.



Hình 3.4: Hình minh họa quá trình đột biến trong GA.

Quá trình tiến hóa sẽ dừng lại khi đạt được một trong các điều kiện sau: tìm được giải pháp thỏa mãn yêu cầu tối thiểu, đạt đến số thế hệ tối đa đã định, hết thời gian hoặc ngân sách tính toán cho phép, hoặc độ thích nghi của giải pháp tốt nhất không còn cải thiện đáng kể qua nhiều thế hệ.

Toàn bộ thuật toán có thể được tóm tắt như trong Hình 3.5:



Hình 3.5: Lưu đồ thuật toán GA.

3.6.2 Particle Swarm Optimization

Particle Swarm Optimization [19] (tối ưu bầy đàn hạt) là một thuật toán tối ưu hóa lấy cảm hứng từ hành vi tập thể của các sinh vật như đàn chim, đàn cá khi tìm kiếm nguồn thức ăn trong tự nhiên. Được giới thiệu bởi Kennedy và Eberhart vào năm 1995, PSO thuộc nhóm thuật toán tối ưu hóa ngẫu nhiên, hoạt động bằng cách khai thác thông tin từ nhiều cá thể trong không gian tìm kiếm để tìm ra lời giải tốt nhất. Với đặc điểm đơn giản, dễ triển khai và hiệu quả cao, PSO được áp dụng rộng rãi trong các bài toán tối ưu hóa liên tục và rời rạc.

PSO dựa trên nguyên tắc mỗi cá thể trong quần thể (gọi là hạt - particle) đại diện cho một giải pháp tiềm năng của bài toán. Mỗi hạt di chuyển trong không gian tìm kiếm với một vận tốc xác định, thể hiện hướng bay về phía các giải pháp khác, và chịu ảnh hưởng bởi hai yếu tố chính: kinh nghiệm cá nhân, tức vị trí tốt nhất mà hạt từng đạt được, và kinh nghiệm tập thể, tức vị trí tốt nhất được tìm thấy bởi toàn bộ quần thể. Dựa trên hai yếu tố này, hạt điều chỉnh vận tốc và vị trí để dần hội tụ về giải pháp tối ưu.

Quá trình tối ưu hóa bằng PSO bắt đầu với việc khởi tạo ngẫu nhiên một tập hợp các hạt trong không gian tìm kiếm, sau đó gán vận tốc ban đầu cho từng hạt và xác định giá trị hàm mục tiêu tại mỗi vị trí. Trong vòng lặp tối ưu hóa, vận tốc của mỗi hạt được cập nhật theo công thức:

$$v_i^{t+1} = w \times v_i^t + c_1 \times r_1 \times (p_i - x_i^t) + c_2 \times r_2 \times (g - x_i^t) \quad (3.14)$$

Trong đó, w là hệ số quán tính, giúp kiểm soát mức độ ảnh hưởng giữa tìm kiếm toàn cục và tìm kiếm cục bộ; c_1, c_2 là các hằng số, lần lượt được gọi là trọng số cá nhân và trọng số xã hội, giúp điều chỉnh mức độ ảnh hưởng của kinh nghiệm cá nhân và tập thể, thường được đặt là 2; r_1, r_2 là các số là các số ngẫu nhiên tuân theo phân phối đều trong khoảng $[0, 1]$; p_i là vị trí tốt nhất từng đạt được của hạt; g là vị trí tốt nhất của toàn

bộ quần thể. Sau khi cập nhật vận tốc, vị trí của hạt được cập nhật theo công thức:

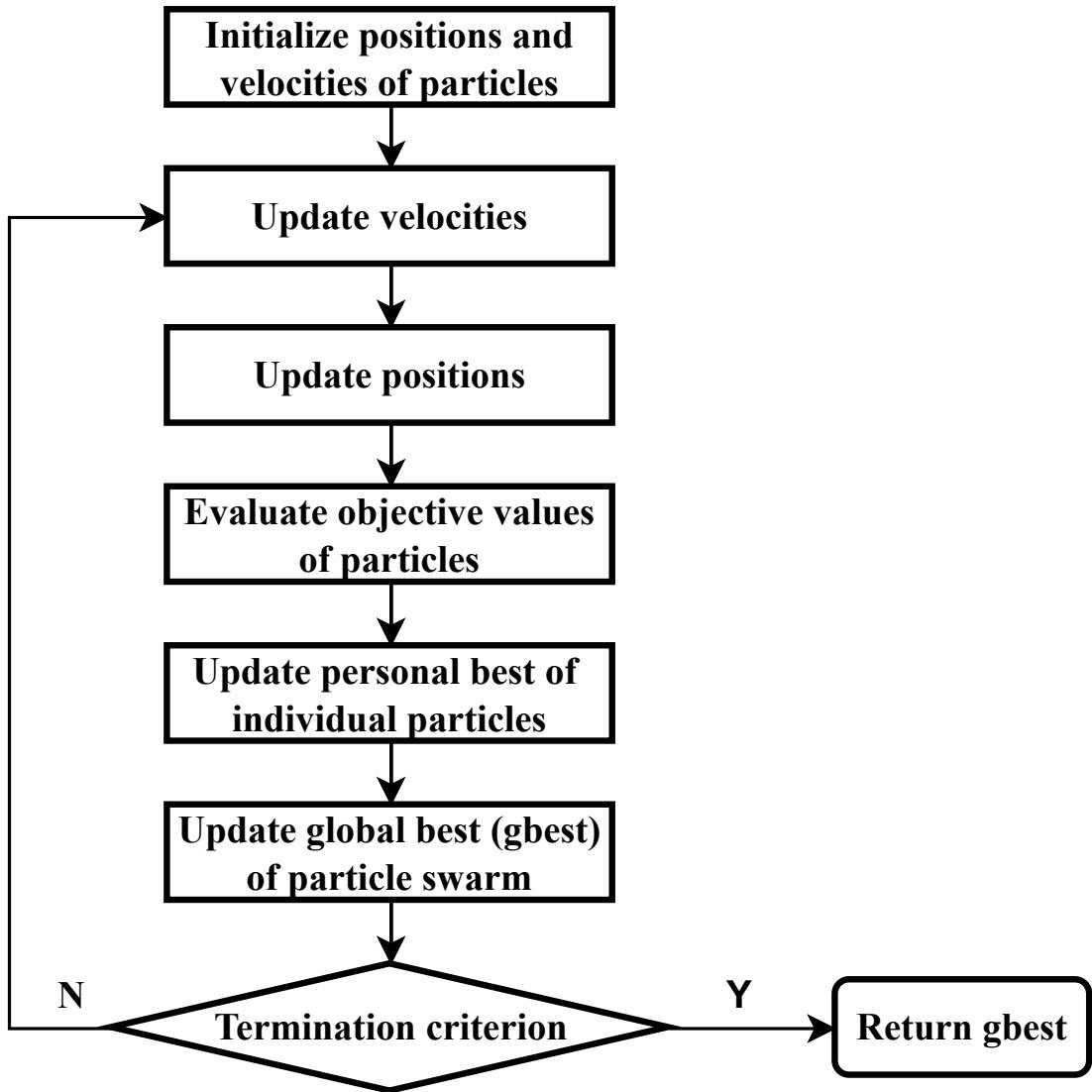
$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (3.15)$$

Nếu giá trị hàm mục tiêu tại vị trí mới tốt hơn giá trị tại p_i , thì p_i được cập nhật. Nếu giá trị hàm mục tiêu của hạt tốt nhất trong quần thể tốt hơn g , thì g cũng được cập nhật. Thuật toán dừng khi đạt số vòng lặp tối đa hoặc khi giá trị hàm mục tiêu không được cải thiện đáng kể sau một số vòng lặp nhất định. Sau khi thuật toán kết thúc, vị trí tốt nhất của toàn bộ quần thể g được xem là lời giải của bài toán. Luồng thực thi của thuật toán được thể hiện trong Hình 3.6.

Ban đầu, PSO được thiết kế cho các bài toán tối ưu hóa liên tục. Tuy nhiên, trong nhiều bài toán thực tế như tối ưu tổ hợp, lập lịch, chọn đặc trưng, lời giải phải là các giá trị rời rạc. Để mở rộng khả năng áp dụng của PSO cho các bài toán rời rạc, vào năm 1997, Kennedy và Eberhart đã giới thiệu Discrete PSO (DPSO) [24].

Trong DPSO, vị trí của hạt được biểu diễn dưới dạng các giá trị rời rạc, chẳng hạn như dãy nhị phân hoặc tập hợp chỉ số. Do đó, khác với PSO liên tục, trong không gian rời rạc, vị trí của hạt không còn là một vector số thực mà có thể là một chuỗi bit, một tập hợp hoặc một hoán vị. Điều này dẫn đến sự thay đổi trong cách định nghĩa vận tốc. Thay vì là một vector số thực, vận tốc trong DPSO thường được diễn giải dưới dạng xác suất hoặc một toán tử biến đổi.

Một trong những phương pháp phổ biến để cập nhật trạng thái của hạt trong DPSO là sử dụng xác suất chuyển đổi. Cụ thể, mỗi phần tử trong nghiệm có một xác suất thay đổi trạng thái dựa trên vận tốc hiện tại. Một phương pháp khác là sử dụng toán tử hoán vị hoặc đột biến, trong đó vận tốc xác định xác suất thực hiện các phép biến đổi như hoán đổi hai phần tử, đảo ngược một chuỗi con hoặc thêm/bớt một phần tử vào tập hợp lời giải.



Hình 3.6: Lưu đồ thuật toán PSO.

Nhờ những điều chỉnh linh hoạt này, PSO nói chung và DPSO nói riêng đã chứng minh hiệu quả trong nhiều bài toán như lập lịch CPU, tối ưu mạng cảm biến, thiết kế lô trình robot và nhiều bài toán tối ưu tổ hợp khác. Với sự phát triển mạnh mẽ trong nghiên cứu và ứng dụng thực tế, PSO và DPSO tiếp tục là các phương pháp tối ưu hóa quan trọng trong nhiều lĩnh vực khoa học và công nghiệp.

3.6.3 Ant Colony Optimization

Ant Colony Optimization [18] (tối ưu đàm kiến) là một thuật toán tối ưu hóa lấy cảm hứng từ hành vi tìm kiếm đường đi của đàm kiến trong tự nhiên. Marco Dorigo lần đầu tiên đề xuất thuật toán này vào năm 1992 dựa trên quan sát rằng các con kiến khi di chuyển sẽ để lại dấu vết pheromone, một chất hóa học giúp dẫn đường cho các con kiến khác. Cơ chế này tạo ra một hình thức giao tiếp gián tiếp giữa các cá thể trong đàm, giúp chúng nhanh chóng tìm ra lộ trình tối ưu đến nguồn thức ăn. Khi nhiều con kiến đi qua một con đường, nồng độ pheromone trên đường đó tăng lên, làm tăng khả năng các con kiến khác lựa chọn con đường này. Đồng thời, pheromone cũng dần bay hơi theo thời gian, giúp loại bỏ các đường ít hiệu quả và tạo cơ hội cho đàm kiến khám phá những con đường tốt hơn. Ý tưởng này đã được chuyển hóa thành một thuật toán tìm kiếm tối ưu, trong đó các "kiến ảo" hoạt động trên một không gian tìm kiếm và dần dần hội tụ đến lời giải tối ưu thông qua việc cập nhật pheromone và sử dụng thông tin heuristic.

Mô hình hoạt động của ACO dựa trên việc mô phỏng cách kiến di chuyển trên một đồ thị, nơi mỗi nút đại diện cho một trạng thái của bài toán và các cạnh thể hiện các bước chuyển đổi giữa các trạng thái đó. Trong quá trình tìm kiếm, mỗi con kiến xây dựng một lời giải bằng cách di chuyển từ nút này sang nút khác dựa trên xác suất. Xác suất này phụ thuộc vào nồng độ pheromone và thông tin heuristic của các cạnh kết nối. Cụ thể, xác suất để một con kiến di chuyển từ nút i đến nút j được tính theo công thức:

$$P_{ij} = \frac{(\tau_{ij})^\alpha \cdot (\eta_{ij})^\beta}{\sum_{k \in N_i} (\tau_{ik})^\alpha \cdot (\eta_{ik})^\beta} \quad (3.16)$$

trong đó, τ_{ij} là lượng pheromone trên cạnh (i, j) , η_{ij} là thông tin heuristic thể hiện mức độ hấp dẫn của cạnh đó (thường là nghịch đảo khoảng cách hoặc chi phí giữa hai nút), còn α và β là các tham số điều chỉnh tầm

quan trọng của pheromone và thông tin heuristic. Tập \mathcal{N}_i gồm tất cả các nút kề mà con kiến có thể di chuyển đến từ nút i .

Sau khi tất cả các con kiến hoàn thành quá trình tìm kiếm trong một vòng lặp, thuật toán sẽ tiến hành cập nhật lại lượng pheromone trên các cạnh để phản ánh chất lượng của các giải pháp được tìm thấy. Cơ chế cập nhật pheromone được thực hiện dựa trên công thức:

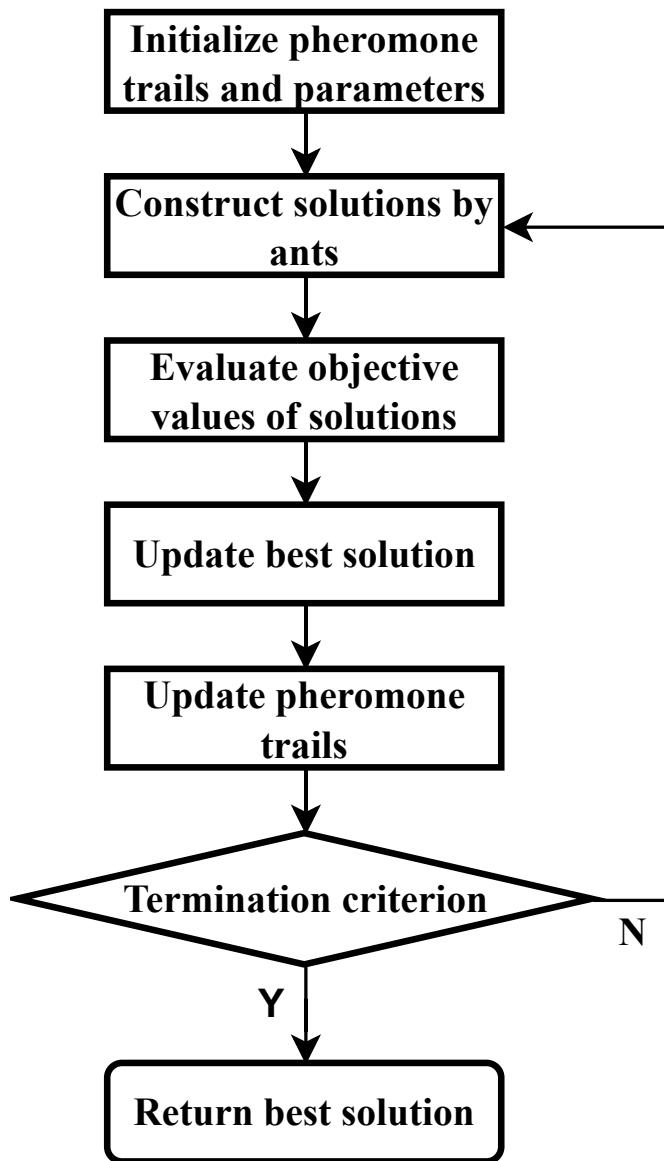
$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^{(k)} \quad (3.17)$$

trong đó, $(1 - \rho) \times \tau_{ij}$ thể hiện quá trình bay hơi pheromone, với ρ là tốc độ bay hơi ($0 < \rho < 1$), và $\Delta\tau_{ij}^{(k)}$ là lượng pheromone do con kiến thứ k đóng góp, được tính theo biểu thức:

$$\Delta\tau_{ij}^{(k)} = \begin{cases} \frac{Q}{L_k}, & \text{nếu kiến } k \text{ đi qua cạnh } (i, j) \\ 0, & \text{ngược lại} \end{cases} \quad (3.18)$$

trong đó, Q là hằng số pheromone và L_k là chất lượng của giải pháp mà kiến thứ k tìm được, thường được đo bằng hàm mục tiêu của bài toán. Quá trình xây dựng giải pháp và cập nhật pheromone này tiếp tục lặp lại qua nhiều vòng cho đến khi thuật toán hội tụ hoặc đạt đến số vòng lặp tối đa. Quá trình hoạt động của thuật toán ACO có thể được mô tả trực quan như trong Hình 3.7.

Thuật toán ACO đã được áp dụng rộng rãi cho nhiều bài toán tối ưu tổ hợp phức tạp, tiêu biểu như bài toán người du lịch (TSP), bài toán định tuyến phương tiện, bài toán lập lịch và bài toán phân công công việc. Nhờ khả năng tìm kiếm song song, thích nghi tốt với môi trường động và khả năng kết hợp linh hoạt với các thuật toán khác, ACO trở thành một công cụ mạnh mẽ trong lĩnh vực tối ưu hóa. Tuy nhiên, thuật toán cũng có những hạn chế như thời gian hội tụ có thể chậm, phụ thuộc nhiều vào tham số và dễ mắc kẹt ở cực trị địa phương.



Hình 3.7: Lưu đồ thuật toán ACO.

Chương 4

Thuật toán đề xuất SO2DI

Chương này trình bày chi tiết thuật toán SO2DI, một phương pháp mới kết hợp chiến lược xóa item và tối ưu hóa ngẫu nhiên nhằm bảo vệ tính riêng tư trong khai thác mẫu hữu ích. Để minh họa quy trình thực thi, chương đưa ra một ví dụ cụ thể, giúp người đọc dễ dàng hình dung cách thuật toán hoạt động.

Thuật toán được đề xuất áp dụng mô hình nhiều dựa trên item để ẩn các SHUI. SO2DI thực hiện xử lý lần lượt từng SHUI, đảm bảo rằng giá trị lợi ích của chúng giảm xuống dưới ngưỡng lợi ích tối thiểu. Với mỗi SHUI s , thuật toán thực hiện năm bước chính:

1. Kiểm tra s đã bị ẩn hay chưa.
2. Xác định item mục tiêu.
3. Xác định transaction mục tiêu.
4. Làm nhiều cơ sở dữ liệu bằng cách xóa item mục tiêu khỏi các transaction mục tiêu.
5. Cập nhật lợi ích cho các NSHUI sau khi thực hiện nhiều.

4.1 Kiểm tra xem SHUI đã bị ẩn hay chưa

Đối với mỗi SHUI, thuật toán thực hiện làm nhiều cơ sở dữ liệu nhằm ẩn SHUI đó. Trong quá trình này, ngoài việc một số NSHUI có thể bị ẩn nhầm, các SHUI khác cũng có thể bị ảnh hưởng. Vì vậy, trước khi thực hiện các thao tác làm nhiều để ẩn một SHUI, thuật toán cần kiểm tra xem SHUI đó đã bị ẩn hay chưa. Điều này giúp thuật toán tiết kiệm thời gian xử lý và tránh thực hiện các tác vụ nhiều không cần thiết, vốn có thể gây ra thêm tác dụng phụ cho cơ sở dữ liệu.

Để kiểm tra một SHUI s đã bị ẩn hay chưa, thuật toán cần tính toán lợi ích của s trong cơ sở dữ liệu hay tổng các lợi ích của s trong từng transaction. Nếu s chưa bị ẩn, các giá trị này sẽ được sử dụng để xác định tập các transaction nhạy cảm, phục vụ việc lựa chọn transaction mục tiêu.

Mã giả mô tả chi tiết bước đầu tiên của thuật toán SO2DI được trình bày trong Thuật toán 9.

Thuật toán 9 ScanDB

Input: s - the SHUI to hide

D - the original database

Output: u_s - the utility of s in D

TID_s - the list of sensitive transaction IDs

UT_s - the list of utilities of s in each sensitive transaction

- 1: $u_s \leftarrow 0$
 - 2: $TID_s \leftarrow []$
 - 3: $UT_s \leftarrow []$
 - 4: **for** each transaction $T_n \in D$ **do**
 - 5: $ut \leftarrow u(s, T_n)$
 - 6: **if** $ut > 0$ **then**
 - 7: Append $T_n.ID$ to TID_s
 - 8: Append ut to UT_s
 - 9: $u_s \leftarrow u_s + ut$
 - 10: **return** (u_s, TID_s, UT_s)
-

Sau khi tính được lợi ích của s , SO2DI sẽ so sánh lợi ích đó với ngưỡng lợi ích tối thiểu δ . Nếu lợi ích của s nhỏ hơn δ , thuật toán sẽ bỏ qua s và chuyển sang xử lý SHUI tiếp theo. Ngược lại, các bước tiếp theo của thuật toán sẽ được thực hiện.

4.2 Xác định item mục tiêu

Quá trình xác định item mục tiêu đóng vai trò quan trọng trong việc tối ưu hóa hiệu quả của thuật toán. Mục tiêu của bước này là lựa chọn một item thuộc s để xóa, sao cho việc giảm lợi ích của s xuống dưới ngưỡng lợi ích tối thiểu được thực hiện nhanh chóng, đồng thời giảm thiểu tác động tiêu cực đối với các NSHUI. Tiêu chí để lựa chọn item mục tiêu là dựa vào tần suất xuất hiện của mỗi item trong các NSHUI, với ưu tiên chọn item xuất hiện ít nhất. Điều này đảm bảo rằng việc xóa item mục tiêu sẽ ít ảnh hưởng đến các NSHUI, từ đó hạn chế tác dụng phụ của quá trình làm nhiễu.

Bên cạnh việc lựa chọn item mục tiêu, thuật toán còn xác định tập các NSHUI liên quan trực tiếp đến việc xóa item mục tiêu. Đây là những NSHUI bị ảnh hưởng bởi quá trình nhiễu và cần được theo dõi để đánh giá tác động của việc xóa item mục tiêu. Việc chỉ theo dõi tập hợp các NSHUI liên quan này, thay vì toàn bộ NSHUI trong cơ sở dữ liệu, giúp giảm thiểu chi phí tính toán, qua đó tối ưu hóa hiệu suất của thuật toán.

Cụ thể, để thực hiện bước này, thuật toán trước tiên duyệt qua từng item thuộc s và tìm tất cả các NSHUI mà item đó xuất hiện. Sau đó, tần suất xuất hiện của mỗi item trong các NSHUI được tính toán và lưu trữ. Dựa trên thông tin thu được, item mục tiêu được chọn là item có số lượng NSHUI liên quan ít nhất. Đồng thời, tập các NSHUI liên quan đến item mục tiêu cũng được xác định để hỗ trợ các bước xử lý tiếp theo.

Chi tiết quá trình được mô tả trong Thuật toán 10.

Thuật toán 10 GetVictimItem

Input: s - the SHUI to hide

U_{ns} - the dictionary mapping each NSHUI to its utility in D

Output: i_v - the victim item selected from s

R - the set of NSHUIs containing i_v

```
1:  $A \leftarrow \{\}$ 
2: for each item  $i_m \in s$  do
3:    $A[i_m] \leftarrow []$ 
4: for each item  $i_m \in s$  do
5:   for each itemset  $X \in U_{ns}$  do
6:     if  $i_m \in X$  then
7:       Append  $X$  to  $A[i_m]$ 
8:  $i_v \leftarrow \operatorname{argmin}_{i_m \in s} |A[i_m]|$ 
9:  $R \leftarrow A[i_v]$ 
10: return  $(i_v, R)$ 
```

4.3 Xác định transaction mục tiêu

Sau khi đã xác định được item mục tiêu, bước tiếp theo của thuật toán là lựa chọn transaction mục tiêu. Đây là những transaction mà item mục tiêu sẽ bị xóa để giảm lợi ích của s xuống dưới ngưỡng lợi ích tối thiểu. Việc lựa chọn này đòi hỏi sự cân nhắc kỹ lưỡng để vừa bảo vệ tính riêng tư vừa giảm thiểu những tác dụng phụ.

Đây là một bài toán tối ưu đa mục tiêu với hai mục tiêu được cân nhắc đồng thời là: (1) giảm lợi ích của s xuống dưới ngưỡng lợi ích tối thiểu và (2) giảm thiểu số lượng NSHUI bị ẩn nhầm; tương ứng với hai tác dụng phụ là Hiding Failure và Missing Cost. Thuật toán không xem xét tác dụng phụ Artificial Cost vì phương pháp làm nhiễu dựa trên việc xóa item không làm tăng lợi ích của bất kỳ itemset nào.

4.3.1 Mô hình hóa bài toán

Quá trình xác định transaction mục tiêu bắt đầu từ tập hợp các transaction nhạy cảm, tức là những transaction mà s xuất hiện và đóng góp trực tiếp vào lợi ích của nó. Các transaction này được xác định dựa trên giá trị lợi ích của s trong từng transaction, vốn đã được tính toán và lưu trữ từ bước kiểm tra SHUI.

Gọi S_{TID} là tập hợp các TID của các transaction nhạy cảm và $V_{TID} \subseteq S_{TID}$ là tập hợp các TID của các transaction mục tiêu cần tìm. Mục tiêu của bài toán là tìm V_{TID}^* , tập hợp tối ưu các TID sao cho sau khi làm nhiều cơ sở dữ liệu, lợi ích của s giảm xuống dưới ngưỡng lợi ích tối thiểu, đồng thời số lượng các NSHUI bị ảnh hưởng là nhỏ nhất. Bài toán này được mô tả thông qua công thức tối ưu hóa sau:

$$V_{TID}^* = \operatorname{argmin}_{V_{TID} \subseteq S_{TID}} f(V_{TID}) \quad (4.1)$$

với $f(V_{TID})$ là hàm mục tiêu đánh giá mức độ phù hợp của V_{TID} .

4.3.2 Hàm mục tiêu

Hàm mục tiêu trong bài toán này được thiết kế nhằm đánh giá đồng thời hai mục tiêu: (1) giảm lợi ích của s xuống dưới ngưỡng lợi ích tối thiểu và (2) giảm thiểu số lượng NSHUI bị ẩn nhầm. Trong thuật toán PPUMGAT, hàm mục tiêu được biểu diễn dưới dạng tổng trọng số của ba tác dụng phụ:

$$f_{PPUMGAT} = w_1 \times \alpha + w_2 \times \beta + w_3 \times \gamma \quad (4.2)$$

trong đó:

- α là số lượng SHUI không bị ẩn sau quá trình làm nhiễu.
- β là số lượng NSHUI bị ẩn nhầm.

- γ là số lượng HUI nhân tạo được tạo ra sau quá trình làm nhiễu.
- w_1, w_2, w_3 là các trọng số thể hiện mức độ quan trọng của từng tác dụng phụ.

Với thuật toán đề xuất SO2DI, do phương pháp làm nhiễu dựa trên việc xóa item không tạo ra HUI nhân tạo nên γ luôn bằng 0. Vì vậy, nếu sử dụng hàm mục tiêu trên cho SO2DI thì nó sẽ được đơn giản hóa thành:

$$f_{PPUMGAT} = w_1 \times \alpha + w_2 \times \beta \quad (4.3)$$

trong đó w_1 thường được đặt lớn hơn w_2 nhằm ưu tiên mục tiêu chính là ẩn s . Tuy nhiên, việc lựa chọn các trọng số w_1 và w_2 không phải lúc nào cũng đơn giản và có thể dẫn đến sự thiếu nhất quán.

Để khắc phục điều này, một hàm mục tiêu mới được thiết kế riêng cho thuật toán SO2DI:

$$f_{SO2DI} = \mathbb{I}(u'(s) \geq \delta) \times (|R| + 1) + \beta \quad (4.4)$$

trong đó:

- $u'(s)$ là lợi ích của s sau quá trình làm nhiễu.
- R là tập các NSHUI liên quan đến item mục tiêu.

Vì $|R|$ luôn lớn hơn hoặc bằng β , hàm f_{SO2DI} đảm bảo rằng mục tiêu ẩn s luôn được ưu tiên cao hơn so với việc giảm số lượng NSHUI bị ẩn nhầm.

Xét itemset X và transaction T_n : Nếu $X \not\subseteq T_n$ thì $u(X, T_n) = 0$. Do đó, nếu xóa item i_m của X khỏi T_n thì lợi ích của X giảm còn $u(X) - u(X, T_n)$. Dựa vào điều này, $u'(s)$ và β có thể được biểu diễn theo V_{TID} như sau:

$$u'(s) = u(s) - \sum_{q \in V_{TID}} u(s, T_q) \quad (4.5)$$

$$\beta = \sum_{X \in R} \mathbb{I} \left(u(X) - \sum_{q \in V_{TID}} u(X, T_q) < \delta \right) \quad (4.6)$$

Mã giả hàm mục tiêu được trình bày ở Thuật toán 11.

Thuật toán 11 ComputeObjective

Input: V - the set of selected victim transaction IDs

δ - the minimum utility threshold

u_s - the utility of s in D

TID_s - the list of sensitive transaction IDs

UT_s - the list of utilities of s in each sensitive transaction

R - the set of NSHUIs containing i_v

U_{ns} - the dictionary mapping each NSHUI to its utility in D

UT_{ns} - the dictionary mapping each NSHUI to its list of utilities in each sensitive transaction

Output: The value of the objective function

```

1:  $ur \leftarrow u_s$ 
2: for each ID  $q \in V$  do
3:    $idx \leftarrow TID_s.index(q)$ 
4:    $ur \leftarrow ur - UT_s[idx]$ 
5:    $I \leftarrow \begin{cases} 1 & \text{if } ur \geq \delta \\ 0 & \text{otherwise} \end{cases}$ 
6:    $\beta \leftarrow 0$ 
7: for each itemset  $X \in R$  do
8:    $u_X \leftarrow U_{ns}[X]$ 
9:   for each ID  $q \in V$  do
10:     $idx \leftarrow TID_s.index(q)$ 
11:     $u_X \leftarrow u_X - UT_{ns}[X][idx]$ 
12:    if  $u_X < \delta$  then
13:       $\beta \leftarrow \beta + 1$ 
14: return  $I \times (|R| + 1) + \beta$ 

```

4.3.3 Xác định số lượng transaction mục tiêu tối đa

Trong quá trình lựa chọn các transaction mục tiêu để làm nhiễu nhầm ẩn SHUI, việc xác định số lượng tối đa các transaction cần can thiệp là rất quan trọng. Nếu tập transaction mục tiêu có số lượng lớn hơn số lượng transaction mục tiêu tối đa, quá trình làm nhiễu sẽ tạo ra thêm các tác dụng phụ không cần thiết.

Cụ thể, với mỗi SHUI s và tập các transaction nhạy cảm S_{TID} , giá trị lợi ích của s trong từng transaction đã được tính toán và lưu trữ thành một danh sách. Để xác định số lượng transaction mục tiêu tối đa, các giá trị này được sắp xếp theo thứ tự tăng dần và cộng dồn từ giá trị thấp nhất đến giá trị cao nhất. Quá trình cộng dồn dừng lại khi hiệu số giữa lợi ích ban đầu của s và tổng lợi ích đã được cộng dồn nhỏ hơn δ . Số lượng các transaction đã được cộng dồn chính là số lượng transaction mục tiêu tối đa cần thiết để đảm bảo rằng lợi ích của s sau làm nhiễu sẽ nằm dưới ngưỡng δ .

Mã giả cho bước xác định số lượng transaction mục tiêu tối đa được trình bày trong Thuật toán 12.

Thuật toán 12 FindMaxVictimTrans

Input: u_s - the utility of s in D

UT_s - the list of utilities of s in each sensitive transaction

δ - the minimum utility threshold

Output: k - the maximum number of victim transactions

1: Sort UT_s in increasing order

2: $k \leftarrow 1$

3: $cum \leftarrow 0$

4: **for** each $ut \in UT_s$ **do**

5: $cum \leftarrow cum + ut$

6: **if** $(u_s - cum) < \delta$ **then**

7: **break**

8: $k \leftarrow k + 1$

9: **return** k

4.3.4 Giải bài toán bằng phương pháp tối ưu hóa ngẫu nhiên

Sau khi mô hình hóa bài toán thông qua công thức 4.1 và có được hàm mục tiêu f_{SO2DI} , SO2DI giải bài toán xác định các transaction mục tiêu thông qua việc áp dụng một trong các thuật toán tối ưu hóa ngẫu nhiên được trình bày ở Chương 3.6. Việc lựa chọn giải thuật tối ưu hóa ngẫu nhiên cụ thể có thể linh hoạt dựa trên đặc thù của dữ liệu và yêu cầu thực tế.

Quá trình giải bài toán bằng phương pháp tối ưu hóa ngẫu nhiên được thực hiện theo các bước tổng quát như trong Thuật toán 13

Thuật toán 13 OptimizeVictimTrans

Input: TID_s - the list of sensitive transaction IDs

k - the maximum number of victim transactions

$params$ - the parameters required to calculate the objective function

M - the number of initial candidates

i_{max} - the maximum number of iterations

Output: V^* - the set of optimized victim transactions IDs

```
1:  $C \leftarrow \text{GenerateInitialCandidates}(TID_s, M, k)$ 
2:  $i \leftarrow 0$ 
3: while  $i < i_{max}$  do
4:   for each candidate  $c \in C$  do
5:      $c.value \leftarrow \text{ComputeObjective}(c, params)$ 
6:    $C \leftarrow \text{UpdateCandidates}(C)$             $\triangleright$  Apply stochastic operations
7:    $i \leftarrow i + 1$ 
8:  $V^* \leftarrow \text{argmin}_{c \in C} c.value$ 
9: return  $V^*$ 
```

4.4 Làm nhiễu cơ sở dữ liệu

Sau khi đã xác định được tập hợp các transaction mục tiêu, bước tiếp theo là làm nhiễu cơ sở dữ liệu bằng cách xóa item mục tiêu ra khỏi các transaction mục tiêu đó. Cụ thể, quá trình làm nhiễu được thực hiện theo các bước sau:

- Lặp qua từng transaction mục tiêu đã xác định.
- Trong mỗi transaction, xóa item mục tiêu khỏi danh sách các item.

4.5 Cập nhật lợi ích cho các NSHUI

Sau khi thực hiện bước làm nhiễu cơ sở dữ liệu, công việc cuối cùng cần được tiến hành trong quá trình ẩn từng SHUI là cập nhật lợi ích của các NSHUI liên quan. Việc duy trì tính chính xác lợi ích của các NSHUI sau mỗi lần làm nhiễu cơ sở dữ liệu nhằm đảm bảo tính đúng đắn của thuật toán trong các lần làm nhiễu tiếp theo.

Trước tiên, cần xác định xem s có phải là SHUI cuối cùng cần xử lý hay không. Nếu s là SHUI cuối cùng, thì việc cập nhật lợi ích của các NSHUI sẽ không cần thiết vì không còn bước làm nhiễu nào sau đó. Ngược lại, nếu vẫn còn các SHUI khác cần xử lý, việc cập nhật là bắt buộc để duy trì tính chính xác của thuật toán.

Quá trình cập nhật bắt đầu bằng việc tính toán lợi ích mới cho từng NSHUI liên quan. Lợi ích mới của mỗi NSHUI ns được xác định bằng công thức:

$$u'(ns) = u(ns) - \sum_{q \in V_{TID}^*} u(ns, T_q) \quad (4.7)$$

Sau khi tính toán lợi ích mới, bước tiếp theo là loại bỏ các NSHUI không còn hợp lệ khỏi danh sách. Một NSHUI được coi là không hợp lệ nếu lợi ích mới của nó nhỏ hơn ngưỡng lợi ích tối thiểu. Việc loại bỏ này không chỉ giúp duy trì độ chính xác của thuật toán mà còn làm giảm quy mô bài toán, từ đó tối ưu hóa hiệu suất của thuật toán trong các bước tiếp theo. Chi tiết hơn về cách thức thực hiện được trình bày trong Thuật toán 14.

Thuật toán 14 UpdateNSHUI

Input: R - the set of NSHUIs containing i_v

U_{ns} - the dictionary mapping each NSHUI to its utility in D

UT_{ns} - the dictionary mapping each NSHUI to its list of utilities in each sensitive transaction

V^* - the set of optimized victim transactions IDs

TID_s - the list of sensitive transaction IDs

δ - the minimum utility threshold

Output: U'_{ns} - update of U_{ns}

```

1: for each itemset  $X \in R$  do
2:   for each ID  $q \in V^*$  do
3:      $idx \leftarrow TID_s.index(q)$ 
4:      $U_{ns}[X] \leftarrow U_{ns}[X] - UT_{ns}[X][idx]$ 
5:    $U'_{ns} \leftarrow \{\}$ 
6:   for each  $(X, u_X) \in U_{ns}$  do
7:     if  $u_X \geq \delta$  then
8:       Append  $(X, u_X)$  to  $U'_{ns}$ 
9: return  $U'_{ns}$ 

```

Cuối cùng, mã giả cho toàn bộ thuật toán SO2DI được tổng hợp trong Thuật toán 15.

Thuật toán 15 SO2DI

Input: D - the original database

I_S - the set of SHUIs to be hidden

I_H - the set of HUIs mined from D

δ - the minimum utility threshold

M - the number of initial candidates

i_{max} - the maximum number of iterations

Output: D' - the sanitized database

```
1:  $D' \leftarrow \text{Copy}(D)$ 
2:  $U_{ns} \leftarrow \{(X, u_X) \in I_H | X \notin I_S\}$ 
3: for each itemset  $s \in I_S$  do
4:    $(u_s, TID_s, UT_s) \leftarrow \text{ScanDB}(s, D')$ 
5:   if  $u_s < \delta$  then
6:     continue
7:    $(i_v, R) \leftarrow \text{GetVictimItem}(s, U_{ns})$ 
8:    $k \leftarrow \text{FindMaxVictimTrans}(u_s, UT_s, \delta)$ 
9:    $UT_{ns} \leftarrow \{\}$ 
10:  for each itemset  $X \in R$  do
11:     $UT_{ns}[X] \leftarrow []$ 
12:    for each ID  $q \in TID_s$  do
13:      Append  $u(X, D'[q])$  to  $UT_{ns}[X]$ 
14:     $params \leftarrow (\delta, u_s, TID_s, UT_s, R, U_{ns}, UT_{ns})$ 
15:     $V^* \leftarrow \text{OptimizeVictimTrans}(TID_s, k, params, M, i_{max})$ 
16:    for each ID  $q \in V^*$  do
17:      Remove  $i_v$  from  $D'[q]$ 
18:    if  $s$  is not the last element in  $I_S$  then
19:       $U_{ns} \leftarrow \text{UpdateNSHUI}(R, U_{ns}, UT_{ns}, V^*, TID_s, \delta)$ 
20: return  $D'$ 
```

4.6 Ví dụ minh họa

Một ví dụ được đưa ra để minh họa chi tiết từng bước thực thi của thuật toán đề xuất SO2DI. Xét cơ sở dữ liệu transaction trong Bảng 2.1, giả sử rằng $\{AC\}$ và $\{BCE\}$ là hai itemset nhạy cảm cần ẩn đi, được chọn ngẫu nhiên từ các HUI trong Bảng 2.3. Trong suốt quá trình làm nhiều cơ sở dữ liệu, Bảng 4.1 được sử dụng để theo dõi các NSHUI.

Bảng 4.1: Bảng NSHUIs

Itemset	$\{CDE\}$	$\{ACE\}$	$\{C\}$	$\{ABCE\}$	$\{AB\}$
Utility	200	224	224	226	232
Itemset	$\{BCD\}$	$\{CE\}$	$\{BCDE\}$	$\{ABC\}$	$\{BC\}$
Utility	232	242	256	276	312

Bắt đầu với itemset nhạy cảm $\{AC\}$, thuật toán ScanDB xác định lợi ích của $\{AC\}$ là $u_{\{AC\}} = 256$, danh sách ID của các transaction chứa $\{AC\}$ là $TID_{\{AC\}} = [T_1, T_2, T_3, T_7]$, và lợi ích của $\{AC\}$ trong từng transaction này lần lượt là $UT_{\{AC\}} = [68, 80, 64, 44]$. Tiếp theo, thuật toán GetVictimItem xác định item mục tiêu $i_v = A$ cùng với tập các NSHUI liên quan $R = [\{ABCE\}, \{ACE\}, \{ABC\}, \{AB\}]$. Sau đó, thuật toán FindMaxVictimTrans sắp xếp $UT_{\{AC\}}$ theo thứ tự tăng dần và xác định số lượng transaction mục tiêu tối đa là $k = 2$, do $256 - (44 + 64) < 200 = \delta$. Giả sử thuật toán OptimizeVictimTrans xác định tập transaction mục tiêu tối ưu là $V^* = \{T_3\}$. Khi đó, item A sẽ bị xóa khỏi transaction T_3 . Sau khi xóa, lợi ích của $\{AC\}$ giảm xuống còn $u_{\{AC\}} = 256 - 64 = 192$, nhỏ hơn ngưỡng 200, do đó $\{AC\}$ đã được ẩn thành công. Kết thúc quá trình ẩn $\{AC\}$, thuật toán UpdateNSHUI cập nhật lợi ích của các NSHUI liên quan lần lượt là 226, 224, 204, 192; khiến $\{AB\}$ cũng bị ẩn theo. Bảng 4.1 được cập nhật lại thành Bảng 4.2.

Bảng 4.2: Bảng NSHUIs sau khi ẩn $\{AC\}$

Itemset	$\{CDE\}$	$\{ACE\}$	$\{C\}$	$\{ABCE\}$
Utility	200	224	224	226
Itemset	$\{BCD\}$	$\{CE\}$	$\{BCDE\}$	$\{ABC\}$
Utility	232	242	256	204
				312

Tiếp tục với itemset nhạy cảm $\{BCE\}$, thuật toán ScanDB xác định lợi ích của $\{BCE\}$ là $u_{\{BCE\}} = 312$, danh sách các transaction chứa $\{BCE\}$ là $TID_{\{BCE\}} = [T_1, T_2, T_8]$ và lợi ích tương ứng trong các transaction này là $UT_{\{BCE\}} = [72, 110, 130]$. Thuật toán GetVictimItem xác định item mục tiêu $i_v = E$ cùng với tập các NSHUI liên quan $R = [\{ABCE\}, \{ACE\}, \{BCDE\}, \{CE\}, \{CDE\}]$. Sau đó, thuật toán Find-MaxVictimTrans xác định số lượng transaction mục tiêu tối đa là $k = 2$, và thuật toán OptimizeVictimTrans chọn transaction tối ưu $V^* = \{T_8\}$. Sau khi xóa item E khỏi transaction T_8 , lợi ích của $\{BCE\}$ giảm xuống còn $u_{\{BCE\}} = 312 - 130 = 182$, nhỏ hơn ngưỡng 200, do đó $\{BCE\}$ đã được ẩn thành công. Đồng thời, lợi ích của các NSHUI liên quan được thuật toán UpdateNSHUI cập nhật thành 226, 224, 118, 160, 110; làm cho $\{BCDE\}$, $\{CE\}$, $\{CDE\}$ cũng bị ẩn. Bảng 4.2 được cập nhật thành Bảng 4.3.

Bảng 4.3: Bảng NSHUIs sau khi ẩn $\{AC\}$, $\{BCE\}$

Itemset		$\{ACE\}$	$\{C\}$	$\{ABCE\}$
Utility		224	224	226
Itemset	$\{BCD\}$		$\{ABC\}$	$\{BC\}$
Utility	232		204	312

Sau khi hoàn tất quá trình làm nhiễu, cơ sở dữ liệu bị sửa đổi thành D' , được trình bày trong Bảng 4.4. Các SHUI $\{AC\}$ và $\{BCE\}$ đã được ẩn thành công, đồng thời cũng làm mất đi các NSHUI $\{AB\}$, $\{BCDE\}$, $\{CE\}$, $\{CDE\}$. Các chỉ số đánh giá tác dụng của thuật toán SO2DI trên tập dữ liệu này là $HF = 0.0$, $MC = 0.4$ và $AC = 0.0$.

Bảng 4.4: Cơ sở dữ liệu nhiễu D'

TID	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
T_1	5	1	6	0	8
T_2	6	6	7	8	3
T_3	0	1	4	0	0
T_4	4	0	0	8	4
T_5	0	1	0	0	0
T_6	9	7	0	9	0
T_7	5	0	3	6	5
T_8	0	6	8	8	0

Chương 5

Thực nghiệm và đánh giá

Chương này cung cấp kết quả thực nghiệm so sánh thuật toán SO2DI với các thuật toán MSU-MAU, MSU-MIU, FILP và PPUMGAT. Thực nghiệm được thực hiện trên nhiều tập dữ liệu khác nhau để đánh giá thời gian thực thi và mức độ giảm thiểu tác dụng phụ của các thuật toán. Ngoài ra, chương cũng phân tích và đánh giá ưu điểm, nhược điểm của từng thuật toán, làm rõ sự khác biệt trong cách tiếp cận và hiệu quả của chúng. Kết quả cho thấy SO2DI có hiệu suất vượt trội so với các phương pháp trên, đặc biệt trên các tập dữ liệu lớn và có mật độ cao.

5.1 Môi trường thực nghiệm và dữ liệu sử dụng

Trong các thử nghiệm, thuật toán đề xuất SO2DI được so sánh với các thuật toán MSU-MAU, MSU-MIU, FILP và PPUMGAT về thời gian thực thi và khả năng giảm thiểu tác dụng phụ. Tất cả các thuật toán được triển khai bằng Julia 1.10.5 và thực thi trên máy tính có cấu hình Intel(R) Core(TM) i7-12700H CPU 2.70 GHz, RAM 8 GB, chạy trên hệ điều hành Microsoft Windows 11 64 bit.

Bảng 5.1: Đặc điểm của các tập dữ liệu thực nghiệm

Dataset	$ D $	$ I $	$AvgLen$	$Density$
foodmart	4,141	1,559	4.42	0.28
t25i10d10k	9,976	929	24.77	2.67
mushrooms	8,416	119	23.0	19.33
chess	3,196	75	37.0	49.33
retail	88,162	16,470	10.31	0.06
t20i6d100k	99,922	893	19.9	2.23
pumsb	49,046	2,113	74.0	3.5
connect	67,557	129	43.0	33.33

Các thử nghiệm sử dụng tám tập dữ liệu, bao gồm sáu tập dữ liệu thực tế (chess, connect, foodmart, mushrooms, pumsb, retail) và hai tập dữ liệu tổng hợp (t20i6d100k, t25i10d10k). Những tập dữ liệu này được lấy từ SPMF. Bảng 5.1 trình bày các đặc điểm chính của chúng, trong đó $|D|$ là số lượng transaction; $|I|$ là số lượng item riêng biệt; $AvgLen$ là số lượng item trung bình trong một transaction; và $Density$ thể hiện độ thưa thớt của tập dữ liệu, được tính bằng công thức sau:

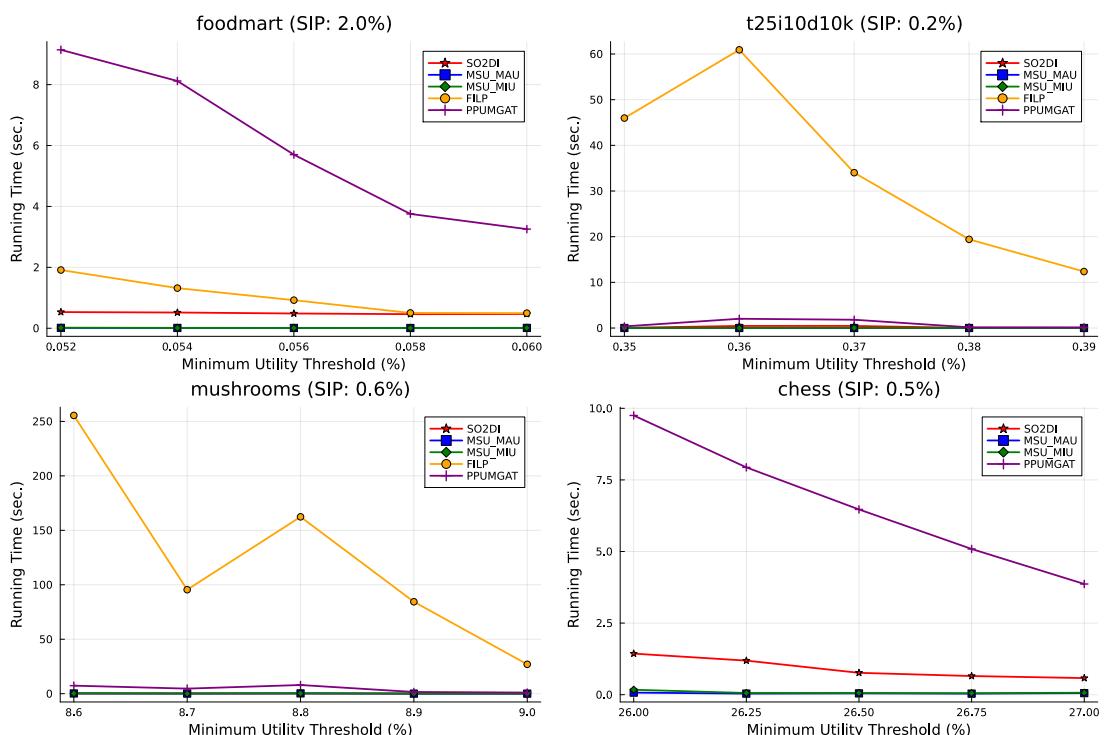
$$Density = \frac{AvgLen}{|I|} \quad (5.1)$$

Do các tập dữ liệu thử nghiệm không chứa lợi ích nội và lợi ích ngoại, giá trị lợi ích được gán theo quy tắc sau. Lợi ích nội được gán cho từng item trong mỗi transaction theo phân phối Uniform trên khoảng [1, 10], trong khi lợi ích ngoại được gán cho từng item riêng biệt theo phân phối Log-Normal trên khoảng [1, 1000]. Các SHUI trong thử nghiệm này được chọn ngẫu nhiên từ tập hợp các HUI, và để khai thác HUI, thuật toán EFIM được sử dụng.

5.2 Kết quả thực nghiệm

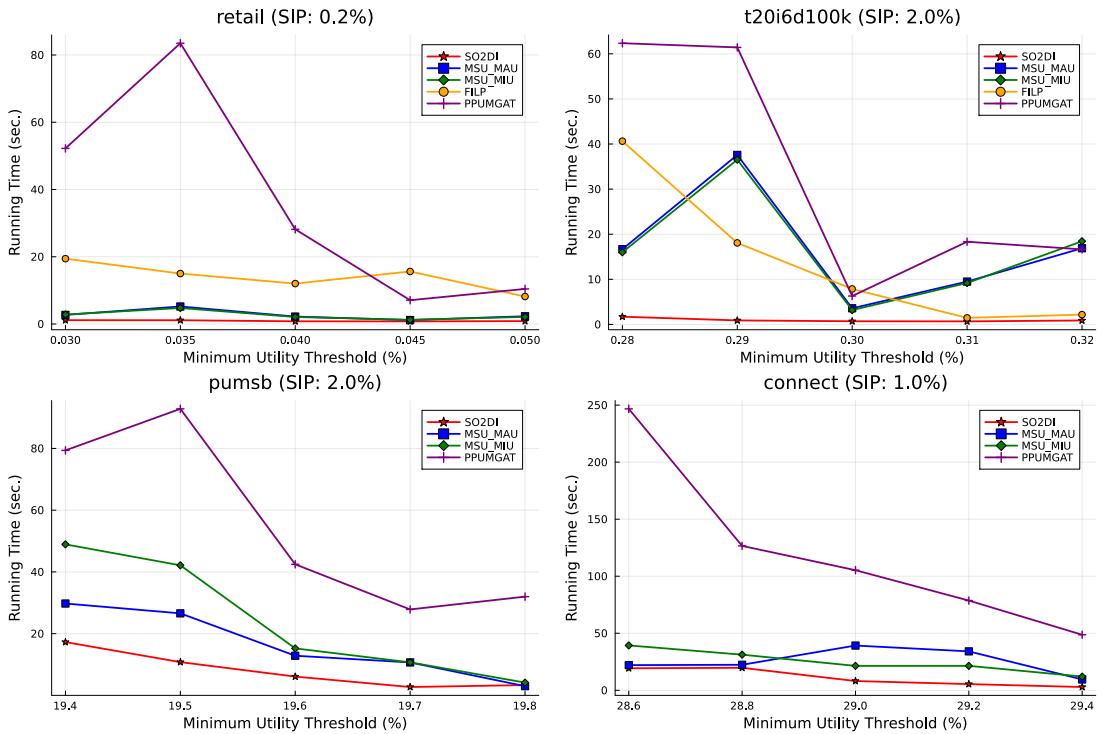
5.2.1 Thời gian thực thi

Phần này trình bày kết quả so sánh thời gian thực thi của năm thuật toán trên các tập dữ liệu khác nhau, với các giá trị ngưỡng lợi ích tối thiểu và tỷ lệ phần trăm thông tin nhạy cảm khác nhau, được minh họa trong Hình 5.1, 5.2, 5.3, và 5.4.



Hình 5.1: So sánh thời gian thực thi trên các tập dữ liệu nhỏ với các ngưỡng lợi ích tối thiểu khác nhau.

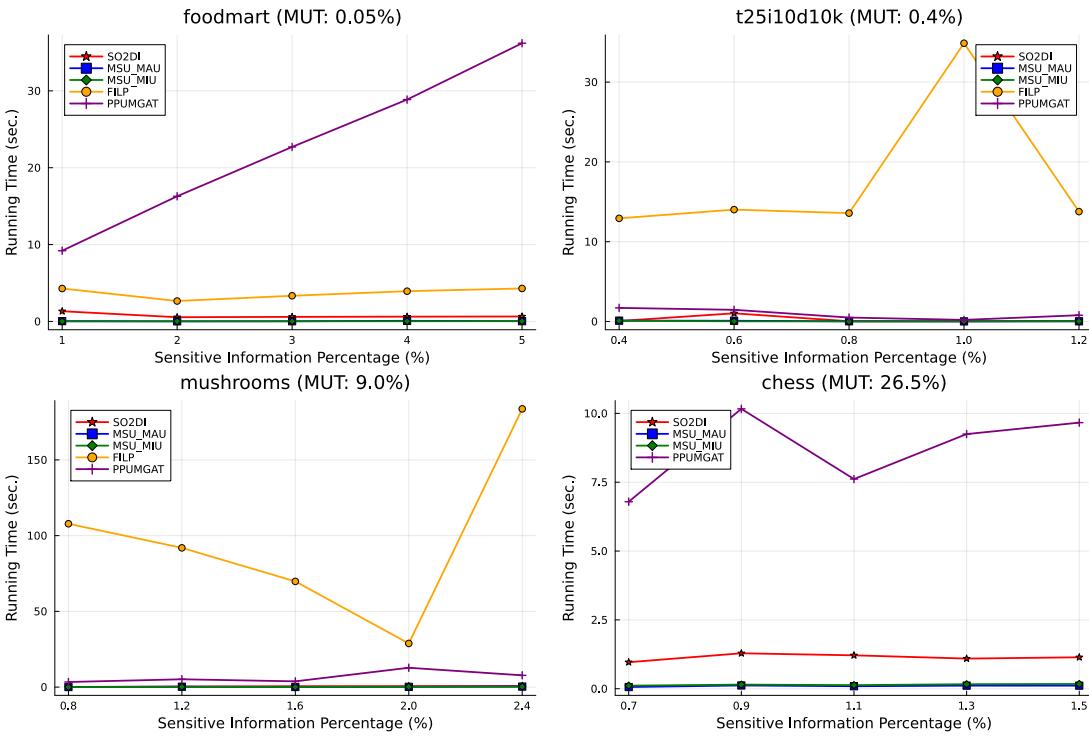
Hình 5.1 và 5.2 cho thấy sự thay đổi thời gian thực thi của các thuật toán khi ngưỡng lợi ích tối thiểu tăng dần, trong đó Hình 5.1 thể hiện kết quả trên bốn tập dữ liệu nhỏ, còn Hình 5.2 thể hiện cho bốn tập dữ liệu lớn. Quan sát cả hai hình, có thể thấy rằng hầu hết các thuật toán đều có thời gian thực thi giảm khi ngưỡng lợi ích tối thiểu δ tăng. Điều này có thể giải thích bởi khi δ tăng, số lượng itemset lợi ích cao giảm, kéo theo số



Hình 5.2: So sánh thời gian thực thi trên các tập dữ liệu lớn với các nguồn lợi ích tối thiểu khác nhau.

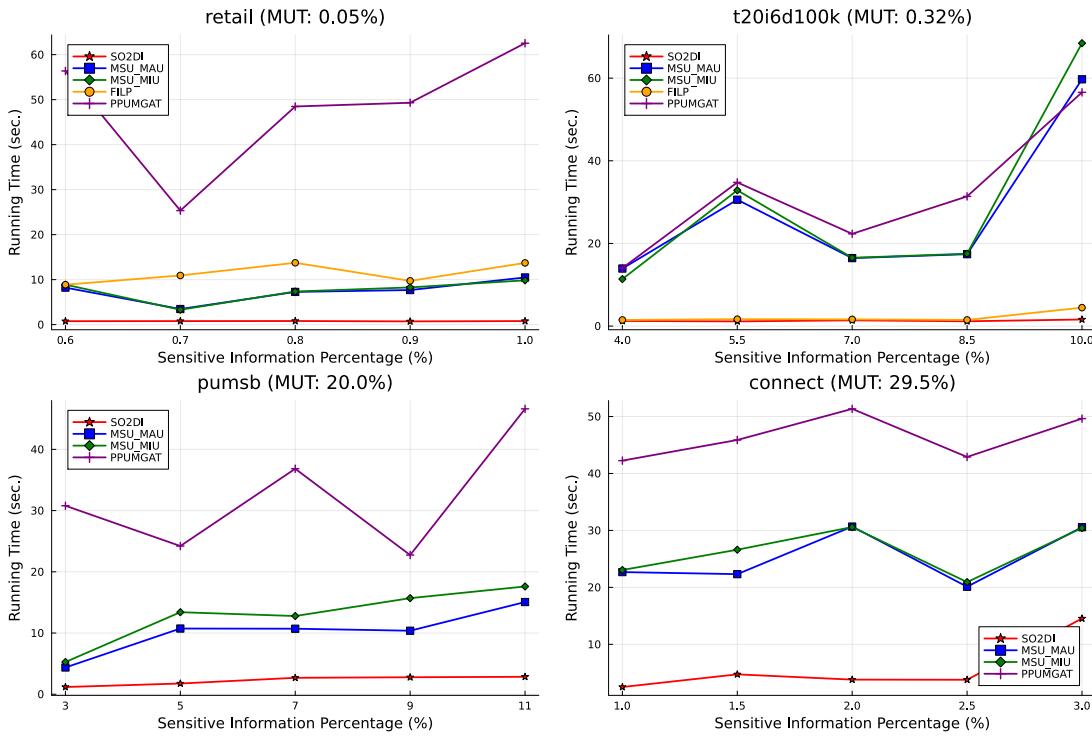
lượng SHUI giảm, dẫn đến việc tính toán và lắp để ẩn SHUI được tối giản, giúp giảm thời gian thực thi. Trên các tập dữ liệu nhỏ, thuật toán đề xuất SO2DI hoạt động kém hơn hai thuật toán heuristic là MSU-MAU và MSU-MIU nhưng lại tốt hơn nhiều so với hai thuật toán FILP và PPUMGAT, đặc biệt khi density của tập dữ liệu tăng. Ngược lại, với các tập dữ liệu lớn, SO2DI thể hiện hiệu suất vượt trội so với tất cả các thuật toán còn lại. Sự khác biệt này xuất phát từ số lần quét cơ sở dữ liệu của từng thuật toán. Trong khi MSU-MAU và MSU-MIU thực hiện nhiều lần quét cơ sở dữ liệu, thì SO2DI, FILP và PPUMGAT chỉ quét một số lần cố định và rất ít. Do đó, khi kích thước tập dữ liệu tăng, thời gian thực thi của các thuật toán heuristic tăng nhanh chóng, khiến SO2DI trở nên hiệu quả hơn.

Tiếp theo, Hình 5.3 và 5.4 thể hiện sự thay đổi thời gian thực thi khi tỷ lệ phần trăm thông tin nhạy cảm tăng dần, trong đó Hình 5.3 áp dụng cho các tập dữ liệu nhỏ, còn Hình 5.4 áp dụng cho các tập dữ liệu lớn. Kết quả cho thấy thời gian thực thi của tất cả các thuật toán đều có xu hướng



Hình 5.3: So sánh thời gian thực thi trên các tập dữ liệu nhỏ với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.

tăng khi tỷ lệ phần trăm thông tin nhạy cảm tăng. Nguyên nhân là khi giữ nguyên ngưỡng lợi ích tối thiểu, việc tăng tỷ lệ thông tin nhạy cảm dẫn đến số lượng SHUI tăng, khiến khối lượng tính toán phải thực hiện cũng lớn hơn. Ở thử nghiệm này, SO2DI tiếp tục thể hiện sự vượt trội so với FILP và PPUMGAT trên mọi tập dữ liệu. Lợi thế này bắt nguồn từ cách tiếp cận khác biệt trong chiến lược ẩn SHUI. Trong khi FILP và PPUMGAT cố gắng ẩn toàn bộ SHUI cùng một lúc, khiến quá trình tính toán trở nên phức tạp, đặc biệt là trên các tập dữ liệu lớn và dày, thì SO2DI ẩn từng SHUI một cách tuần tự, giúp đơn giản hóa tính toán và tối ưu hóa thời gian thực thi. Trong thử nghiệm này, SO2DI vẫn kém hơn MSU-MAU và MSU-MIU trên các tập dữ liệu nhỏ nhưng lại vượt trội trên các tập dữ liệu lớn. Điều này xuất phát từ chiến lược lựa chọn item và transaction mục tiêu. Trên các tập dữ liệu nhỏ, phương pháp heuristic giúp MSU-MAU và MSU-MIU xác định item và transaction mục tiêu nhanh hơn đáng kể so với phương pháp tối ưu hóa ngẫu nhiên của SO2DI. Tuy nhiên, khi kích



Hình 5.4: So sánh thời gian thực thi trên các tập dữ liệu lớn với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.

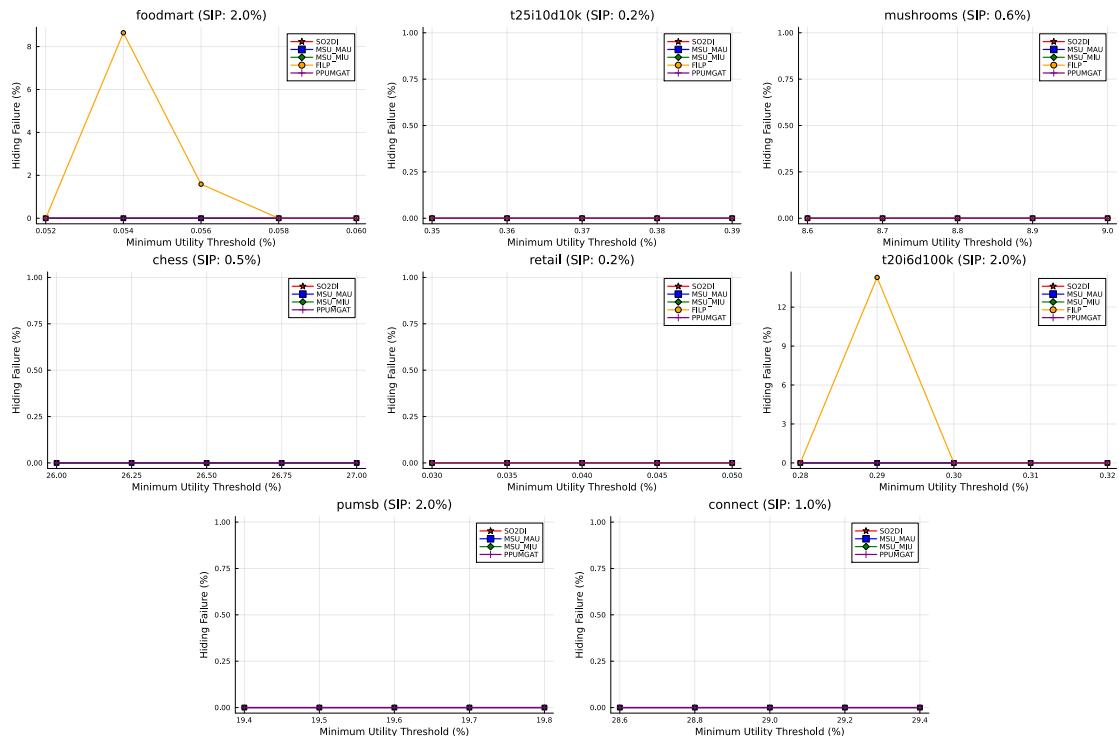
thuộc tập dữ liệu tăng, tối ưu hóa ngẫu nhiên giúp SO2DI lựa chọn các transaction mục tiêu một cách hiệu quả hơn. Kết hợp với chiến lược xóa item, SO2DI có thể ẩn một SHUI nhanh hơn so với MSU-MAU và MSU-MIU, do hai thuật toán heuristic phải giảm lợi ích của SHUI một cách từ từ, làm chậm quá trình xử lý.

Tóm lại, SO2DI chỉ hoạt động kém hơn MSU-MAU và MSU-MIU trên các tập dữ liệu nhỏ, nhưng lại thể hiện sự vượt trội trên các tập dữ liệu lớn, đặc biệt là những tập dữ liệu có độ dày cao như t20i6d100k, pumsb và connect. Kết quả này khẳng định hiệu quả của thuật toán đề xuất SO2DI trong việc xử lý các tập dữ liệu lớn và phức tạp.

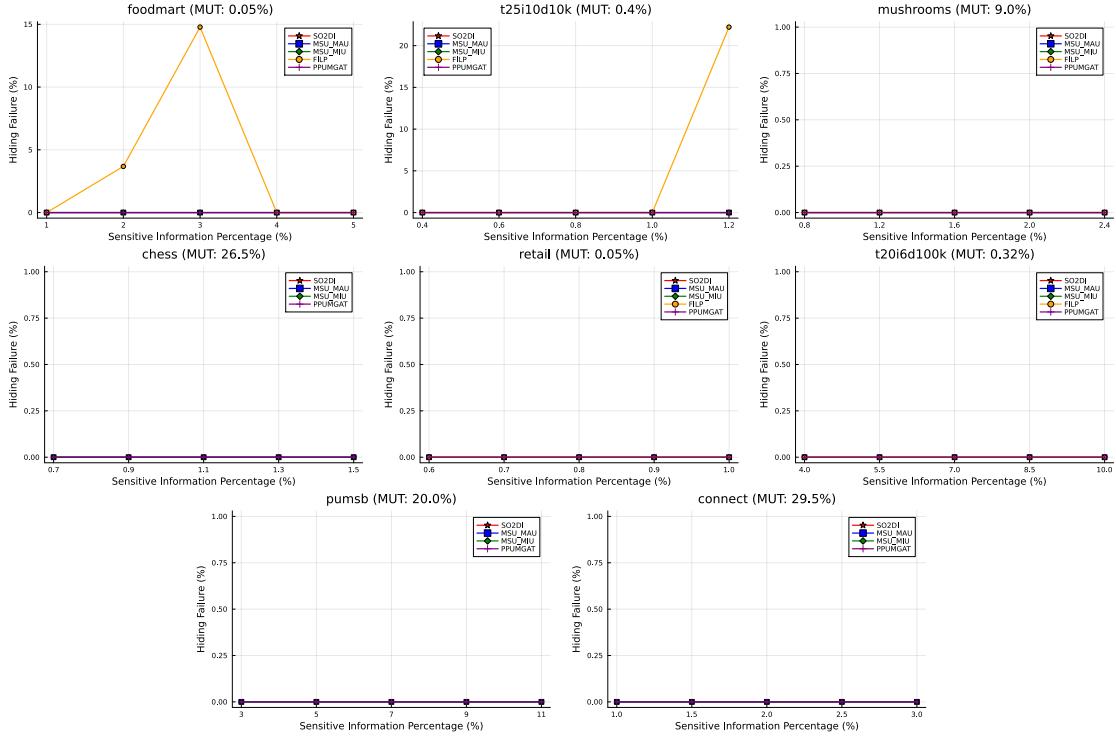
5.2.2 Tác dụng phụ

Trong phần này, ba độ đo đánh giá tác dụng phụ, gồm hiding failure (HF), missing cost (MC) và artificial cost (AC), được sử dụng làm tiêu chí so sánh hiệu quả của thuật toán đề xuất với các thuật toán khác. HF phản ánh tỷ lệ các SHUI không được che giấu thành công sau quá trình làm nhiễu, MC thể hiện tỷ lệ các NSHUI bị vô tình ẩn đi, trong khi AC đo lường tỷ lệ các HUI nhân tạo phát sinh sau khi thực hiện quá trình làm nhiễu. Công thức tính ba độ đo này đã được trình bày chi tiết trong 2.7, 2.8 và 2.9. Các thực nghiệm được tiến hành trên các tập dữ liệu khác nhau với các giá trị ngưỡng lợi ích tối thiểu và tỷ lệ phần trăm thông tin nhạy cảm đa dạng nhằm đảm bảo đánh giá toàn diện về hiệu quả của các thuật toán.

Hiding Failure



Hình 5.5: So sánh Hiding Failure của các thuật toán với các ngưỡng lợi ích tối thiểu khác nhau.



Hình 5.6: So sánh Hiding Failure của các thuật toán với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.

Quan sát Hình 5.5 và Hình 5.6, ta thấy rằng các thuật toán MSU-MAU, MSU-MIU, SO2DI và PPUMGAT đều ẩn giấu thành công tất cả SHUI trên cả tám tập dữ liệu thực nghiệm. Tuy nhiên, thuật toán FILP không đạt được kết quả tương tự và thất bại trong việc ẩn hoàn toàn SHUI trên các tập dữ liệu foodmart, t20i6d100k và t25i10d10k. Nguyên nhân của sự khác biệt này có thể được lý giải dựa trên chiến lược ẩn SHUI của từng thuật toán. Cụ thể, MSU-MAU và MSU-MIU thực hiện quá trình ẩn theo cách tiếp cận lần lượt, nghĩa là với mỗi SHUI, thuật toán sẽ giảm dần lợi ích của nó cho đến khi giá trị này xuống dưới ngưỡng lợi ích tối thiểu δ , sau đó mới tiếp tục xử lý SHUI tiếp theo. Do đó, hai thuật toán này luôn đảm bảo việc ẩn thành công tất cả SHUI trên mọi tập dữ liệu.

Ngược lại, các thuật toán SO2DI và PPUMGAT sử dụng các hàm mục tiêu để đánh giá quá trình ẩn SHUI, trong khi FILP dựa vào hệ thống các ràng buộc của bài toán thỏa mãn ràng buộc (CSP). Việc ẩn SHUI trong FILP chịu ảnh hưởng trực tiếp từ kết quả giải bài toán CSP, do đó thuật

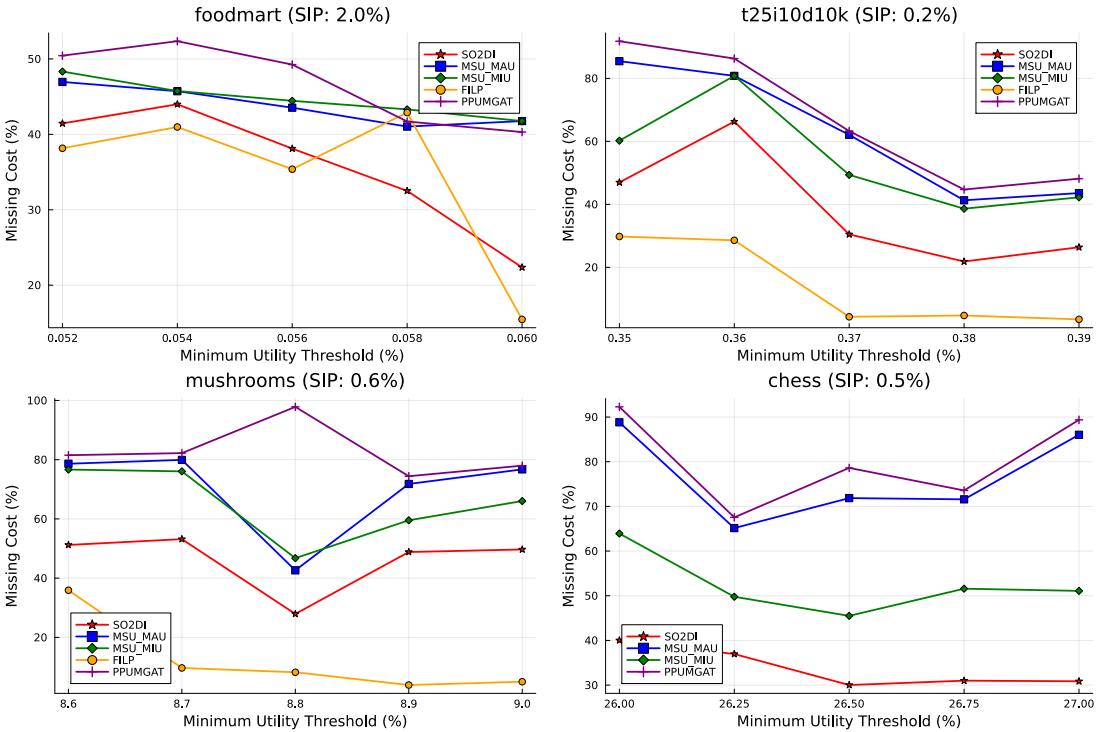
toán này không thể đảm bảo ẩn thành công toàn bộ SHUI trong mọi trường hợp. Trong khi đó, hiệu quả của SO2DI và PPUMGAT phụ thuộc vào cách thiết lập và tối ưu hóa hàm mục tiêu của từng thuật toán. Kết quả thực nghiệm đã chứng minh tính hiệu quả của hai hàm mục tiêu $f_{PPUMGAT}$ và f_{SO2DI} . Tuy nhiên, so với $f_{PPUMGAT}$, hàm mục tiêu của SO2DI thể hiện sự ổn định cao hơn nhờ tính nhất quán trong quá trình tối ưu hóa, trong khi $f_{PPUMGAT}$ vẫn còn phụ thuộc nhiều vào cách lựa chọn trọng số.

Nhìn chung, với tiêu chí hiding failure, SO2DI có thể được đánh giá là một trong những thuật toán mạnh mẽ nhất, chỉ đứng sau hai thuật toán MSU-MAU và MSU-MIU về khả năng đảm bảo ẩn toàn bộ SHUI trên mọi tập dữ liệu. Kết quả thực nghiệm chứng minh rằng SO2DI không chỉ có hiệu suất thực thi tốt mà còn duy trì được độ tin cậy cao trong việc che giấu thông tin nhạy cảm, đặc biệt khi so sánh với PPUMGAT và FILP.

Missing Cost

Hình 5.7 và 5.8 thể hiện sự thay đổi của missing cost theo ngưỡng lợi ích tối thiểu, trong đó Hình 5.7 áp dụng cho bốn tập dữ liệu nhỏ, còn Hình 5.8 áp dụng cho bốn tập dữ liệu lớn. Quan sát cả hai hình, có thể thấy rằng hầu hết các thuật toán đều có xu hướng giảm missing cost khi ngưỡng lợi ích tối thiểu δ tăng. Hiện tượng này tương tự như trong phần phân tích thời gian thực thi. So sánh với MSU-MAU, MSU-MIU và PPUMGAT, SO2DI có missing cost thấp hơn trên cả tám tập dữ liệu thực nghiệm, đặc biệt là khi dữ liệu có độ dày cao.

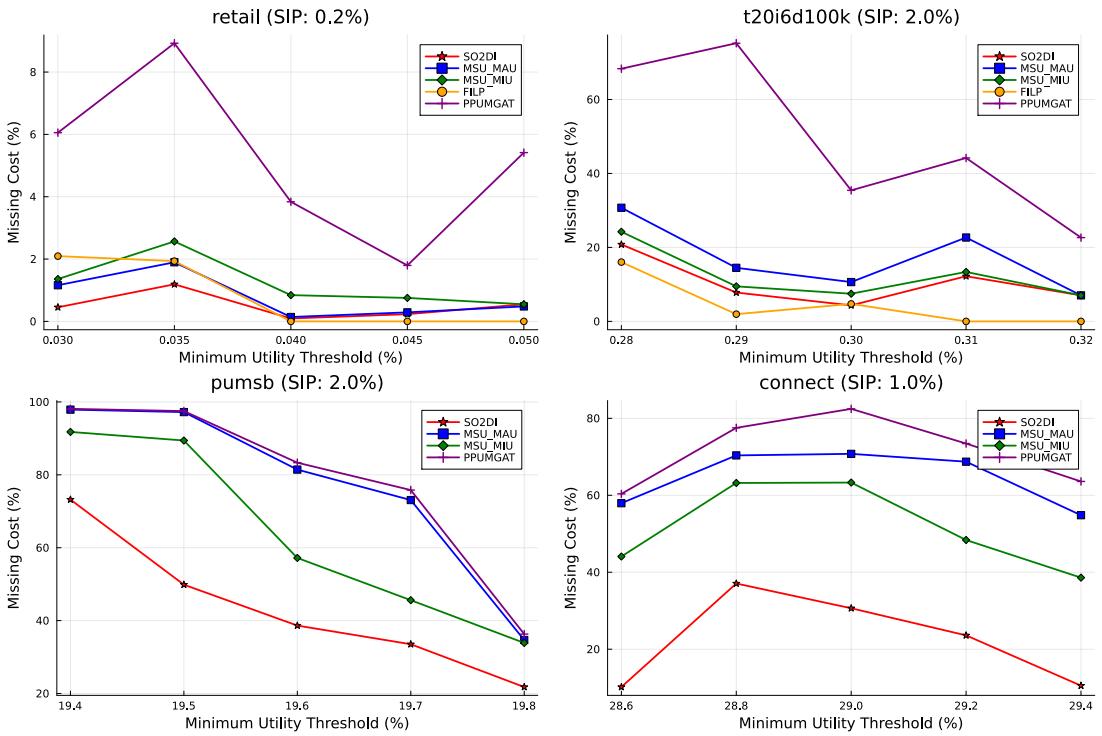
Trên các tập dữ liệu nhỏ, FILP thể hiện sự vượt trội so với SO2DI khi missing cost của nó gần bằng 0%. Tuy nhiên, đối với tập chess, mặc dù có kích thước nhỏ với hơn 3000 transaction, FILP lại không thể chạy thành công do thời gian xử lý quá lâu. Điều này có thể được lý giải bởi mật độ dày đặc của tập chess, với độ density lên đến 49.33%, cao nhất trong số tám tập dữ liệu thực nghiệm. Trong trường hợp này, SO2DI là thuật toán có missing cost thấp nhất trong số các thuật toán được thử nghiệm.



Hình 5.7: So sánh Missing Cost trên các tập dữ liệu nhỏ với các nguồn lợi ích tối thiểu khác nhau.

Trên các tập dữ liệu lớn, FILP không còn duy trì được ưu thế. Ngoại trừ tập t20i6d100k, nơi FILP có missing cost thấp hơn, SO2DI lại thể hiện kết quả tốt hơn trên hầu hết các tập dữ liệu khác. Đặc biệt, trên hai tập dữ liệu lớn và dày là pumsb và connect, FILP không thể chạy thành công, trong khi SO2DI vẫn đạt kết quả ổn định. Điều này cho thấy SO2DI có ưu thế hơn FILP trên các tập dữ liệu lớn, trừ trường hợp t20i6d100k.

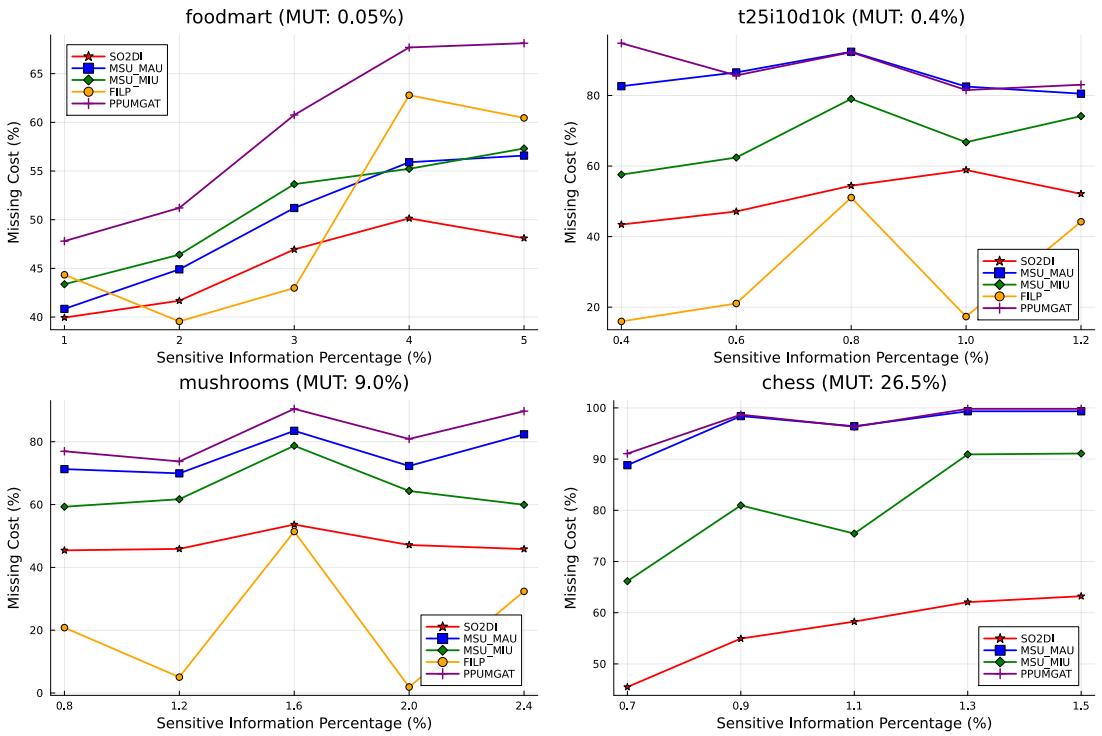
Tiếp theo, Hình 5.9 và 5.10 thể hiện sự thay đổi của missing cost khi tỷ lệ phần trăm thông tin nhạy cảm tăng dần, trong đó Hình 5.9 áp dụng cho các tập dữ liệu nhỏ, còn Hình 5.10 áp dụng cho các tập dữ liệu lớn. Kết quả cho thấy missing cost của tất cả các thuật toán đều có xu hướng tăng khi tỷ lệ thông tin nhạy cảm tăng, tương tự như hiện tượng quan sát được trong thời gian thực thi. Trong thử nghiệm này, SO2DI tiếp tục thể hiện ưu thế so với MSU-MAU, MSU-MIU và PPUMGAT trên tất cả các tập dữ liệu. FILP chỉ vượt trội hơn SO2DI trên ba tập dữ liệu là t25i10d10k, mushrooms và t20i6d100k, trong khi SO2DI cho kết quả tốt hơn trên năm



Hình 5.8: So sánh Missing Cost trên các tập dữ liệu lớn với các ngưỡng lợi ích tối thiểu khác nhau.

tập còn lại.

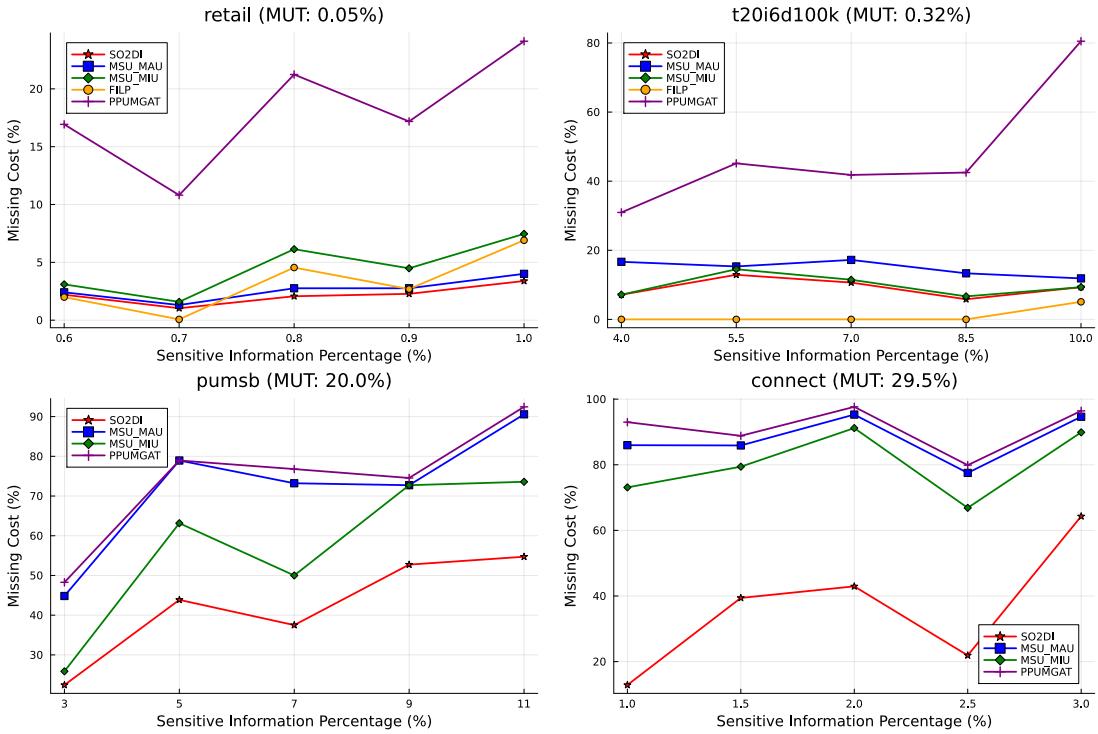
Các kết quả thực nghiệm cho thấy SO2DI tốt hơn rất nhiều so với MSU-MAU, MSU-MIU và PPUMGAT. Missing cost của SO2DI cao nhất cũng chỉ khoảng 60%, trong khi ba thuật toán trên có thể làm mất gần như toàn bộ NSHUI khi tỷ lệ thông tin nhạy cảm cao. Nguyên nhân chính là do MSU-MAU và MSU-MIU chỉ sử dụng các chiến lược heuristic đơn giản để giảm thiểu missing cost. Mặc dù PPUMGAT sử dụng thuật toán di truyền (GA) để tối ưu missing cost, nhưng kết quả của nó lại tệ hơn cả MSU-MAU và MSU-MIU. Nguyên nhân xuất phát từ chiến lược nhiều không hợp lý của PPUMGAT, cụ thể là xóa transaction. Ngay cả khi chọn được tập transaction tối ưu để xóa, chiến lược này vẫn gây mất mát một lượng lớn thông tin không nhạy cảm. Ngược lại, SO2DI sử dụng chiến lược xóa item, giúp giảm đáng kể số lượng NSHUI bị ẩn nhầm.



Hình 5.9: So sánh Missing Cost trên các tập dữ liệu nhỏ với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.

FILP thể hiện sự vượt trội trong việc tối thiểu hóa missing cost trên các tập dữ liệu nhỏ và thừa, khi tỷ lệ mất mát thông tin gần bằng 0%. Tuy nhiên, khi áp dụng trên các tập dữ liệu lớn hoặc dày, thuật toán gặp vấn đề về hiệu suất và không thể chạy thành công. Nguyên nhân chính là do khi độ phức tạp của tập dữ liệu tăng, số lượng ràng buộc cũng tăng theo, khiến bài toán thỏa mãn ràng buộc (CSP) trở nên quá phức tạp.

Tóm lại, SO2DI thể hiện hiệu suất ổn định, vượt trội hơn hẳn MSU-MAU, MSU-MIU và PPUMGAT trong việc giảm thiểu missing cost trên mọi tập dữ liệu thử nghiệm, đồng thời chỉ kém FILP trên các tập dữ liệu nhỏ và thừa. Ngoài ra, SO2DI duy trì được mức missing cost thấp ngay cả trên các tập dữ liệu có kích thước lớn và độ phức tạp cao.

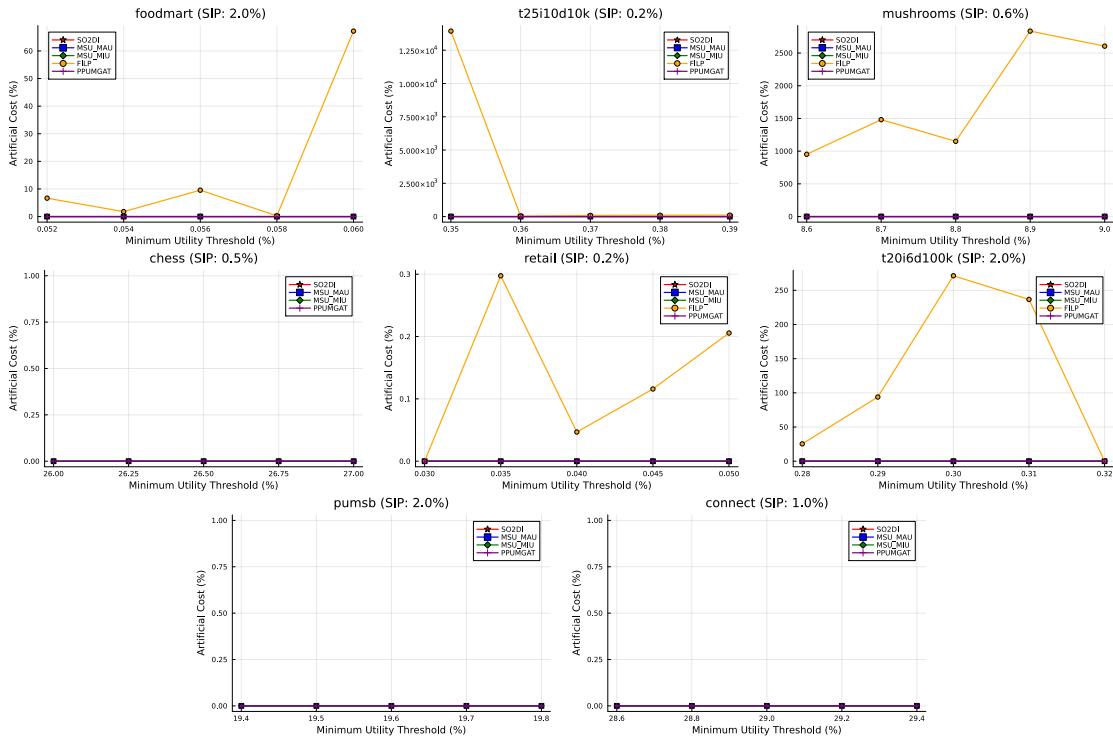


Hình 5.10: So sánh Missing Cost trên các tập dữ liệu lớn với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.

Artificial Cost

Quan sát Hình 5.11 và Hình 5.12 cho thấy các thuật toán MSU-MAU, MSU-MIU, SO2DI và PPUMGAT đều duy trì artificial cost ở mức 0% trên tất cả các tập dữ liệu thử nghiệm. Điều này là do các thuật toán này chỉ giảm internal utility của item, xóa item hoặc xóa transaction nên không tạo ra bất kỳ HUI nhân tạo nào. Ngược lại, FILP sinh ra một lượng lớn HUI nhân tạo trên năm tập dữ liệu còn lại, ngoại trừ ba tập chess, pumsb và connect, nơi thuật toán này không chạy thành công.

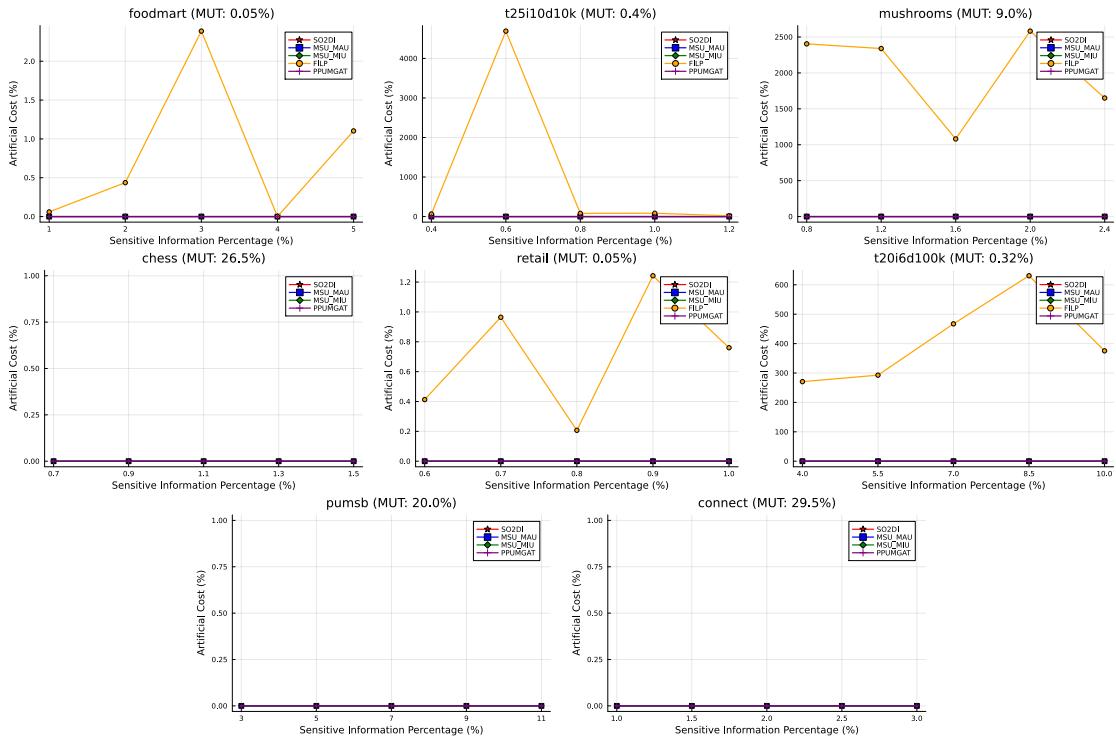
Kết quả thực nghiệm cho thấy artificial cost của FILP không ổn định và không tuân theo một quy luật nhất định. Trong một số trường hợp, giá trị này bằng 0%, nhưng sau đó có thể tăng đột ngột lên trên 4000% rồi lại giảm xuống 0%. Tập retail có kết quả tốt nhất khi artificial cost cao nhất chỉ khoảng 1.2%, trong khi tập mushrooms cho kết quả tệ nhất với artificial cost dao động từ 1000% đến hơn 2500%.



Hình 5.11: So sánh Artificial Cost của các thuật toán với các ngưỡng lợi ích tối thiểu khác nhau.

Những phát hiện trên cho thấy việc FILP có thể đưa missing cost về gần 0% thực chất chỉ là một sự đánh đổi. Quan sát lại phần Missing Cost, có thể thấy rằng ở những trường hợp FILP có missing cost thấp hơn đáng kể so với SO2DI, artificial cost lại tăng đột biến. Điều này cho thấy FILP giảm thiểu mất mát thông tin bằng cách bổ sung một lượng lớn thông tin nhân tạo vào cơ sở dữ liệu.

Tóm lại, SO2DI, MSU-MAU, MSU-MIU và PPUMGAT duy trì artificial cost ở mức 0% trên tất cả các tập dữ liệu, trong khi FILP tạo ra một lượng lớn HUI nhân tạo và có artificial cost không ổn định.



Hình 5.12: So sánh Artificial Cost của các thuật toán với các tỷ lệ phần trăm thông tin nhạy cảm khác nhau.

Từ các kết quả thực nghiệm, có thể thấy rằng SO2DI là một thuật toán cân bằng tốt giữa hiệu suất thực thi và tối thiểu hóa tác dụng phụ. Về thời gian thực thi, SO2DI cho kết quả vượt trội so với FILP và PPUMGAT, đồng thời nhanh hơn đáng kể so với MSU-MAU và MSU-MIU trên các tập dữ liệu lớn. Về tác dụng phụ, SO2DI thể hiện khả năng che giấu thông tin nhạy cảm tốt khi đạt hiding failure bằng 0% trên mọi thử nghiệm. SO2DI cũng vượt trội hơn cả ba thuật toán MSU-MAU, MSU-MIU và PPUMGAT về missing cost. So với FILP, SO2DI có tỷ lệ missing cost cao hơn trên các tập dữ liệu nhỏ và thưa nhưng lại ổn định và tốt hơn trên các tập dữ liệu lớn và dày, trong khi FILP gặp vấn đề khi xử lý những tập dữ liệu này. Ngoài ra, SO2DI không tạo ra HUI nhân tạo, trong khi FILP có artificial cost rất cao trên nhiều tập dữ liệu. Nhìn chung, SO2DI là một thuật toán mạnh mẽ, vừa đảm bảo che giấu thông tin nhạy cảm hoàn toàn, vừa duy trì mức độ tác dụng phụ thấp và hiệu suất thực thi tốt. Điều này khiến SO2DI trở thành một giải pháp tiềm năng cho bài toán PPUM.

Chương 6

Kết luận và hướng phát triển

Chương này tổng kết lại những đóng góp chính của luận văn, khẳng định hiệu quả của thuật toán SO2DI trong việc bảo vệ tính riêng tư mà vẫn duy trì chất lượng khai thác mẫu hữu ích. Đồng thời, chương này đề xuất các hướng phát triển tiềm năng cho nghiên cứu trong tương lai, bao gồm tối ưu hóa thứ tự ẩn các SHUI và cải thiện phương pháp chọn item mục tiêu.

6.1 Kết luận

Bảo vệ tính riêng tư trong khai thác mẫu hữu ích là một bài toán quan trọng, đặt ra thách thức lớn trong việc cân bằng giữa che giấu thông tin nhạy cảm và duy trì tính hữu ích của dữ liệu. Điều này đòi hỏi những phương pháp tiếp cận hiệu quả nhằm hạn chế rủi ro tiết lộ thông tin mà vẫn giảm thiểu tác dụng phụ sau quá trình ẩn dữ liệu. Một trong những hướng tiếp cận đáng chú ý là phương pháp xóa transaction để ẩn thông tin nhạy cảm, được áp dụng trong thuật toán PPUMGAT. Phương pháp này sử dụng thuật toán di truyền để tìm tập transaction mục tiêu tối ưu thay vì dựa vào heuristic như các phương pháp trước đó. Tuy nhiên, dù mang lại những cải tiến đáng kể, PPUMGAT vẫn tồn tại những hạn chế về thời gian thực thi và có thể gây ra các tác dụng phụ không mong muốn.

Trước thực trạng đó, luận văn đã tập trung nghiên cứu các phương pháp bảo vệ tính riêng tư trong khai thác mẫu hữu ích nhằm đề xuất một thuật toán mới khắc phục những nhược điểm của các phương pháp hiện có. Trong quá trình thực hiện, luận văn đã tổng hợp và phân tích các thuật toán khai thác mẫu hữu ích cũng như các phương pháp bảo vệ tính riêng tư phổ biến. Bên cạnh đó, các kỹ thuật tối ưu hóa ngẫu nhiên cũng được nghiên cứu nhằm tìm ra hướng ứng dụng hiệu quả cho bài toán PPUM.

Dựa trên những nghiên cứu này, luận văn đã triển khai và kiểm chứng nhiều thuật toán PPUM, bao gồm MSU-MAU, MSU-MIU, FILP và PPUM-GAT, cũng như thuật toán khai thác mẫu hữu ích EFIM. Đồng thời, các thuật toán tối ưu hóa ngẫu nhiên như GA, PSO và ACO đã được ứng dụng để đánh giá khả năng lựa chọn tập transaction mục tiêu tối ưu, từ đó làm cơ sở cho việc đề xuất thuật toán mới.

Cuối cùng, luận văn đã đề xuất thuật toán SO2DI, một phương pháp kết hợp chiến lược xóa item với tối ưu hóa ngẫu nhiên nhằm cải thiện hiệu quả ẩn dữ liệu nhạy cảm. Kết quả thực nghiệm cho thấy SO2DI không chỉ có hiệu suất vượt trội so với các phương pháp trước đó, đặc biệt trên các tập dữ liệu lớn và dày, mà còn giúp giảm thiểu thiểu đáng kể các tác dụng phụ trong quá trình bảo vệ thông tin. Những kết quả đạt được không chỉ góp phần nâng cao hiệu quả bảo vệ tính riêng tư trong khai thác mẫu hữu ích mà còn mở ra hướng nghiên cứu mới cho các phương pháp bảo vệ thông tin dựa trên tối ưu hóa ngẫu nhiên.

6.2 Hướng phát triển

Từ các kết quả đạt được trong luận văn, có thể thấy rằng vẫn còn nhiều khía cạnh cần tiếp tục nghiên cứu để nâng cao hiệu quả của thuật toán SO2DI cũng như cải thiện các phương pháp bảo vệ tính riêng tư trong khai thác mẫu hữu ích. Một trong những hướng phát triển quan trọng là tối ưu hóa thứ tự ẩn các SHUI. Thí tự này có ảnh hưởng lớn đến thời gian thực thi cũng như tác dụng phụ phát sinh từ quá trình làm nhiễu. Trong luận văn, thuật toán SO2DI sử dụng thứ tự mặc định của tập SHUI mà không có một chiến lược điều chỉnh tối ưu nào. Điều này có thể làm giảm hiệu quả của thuật toán và gây ra những tác dụng phụ không mong muốn. Vì vậy, trong tương lai, cần nghiên cứu các tiêu chí và chiến lược sắp xếp lại thứ tự ẩn SHUI để giảm thiểu chi phí tính toán và tối ưu hóa chất lượng của tập dữ liệu sau khi xử lý.

Bên cạnh đó, việc chọn item mục tiêu trong thuật toán SO2DI hiện tại vẫn dựa trên một phương pháp heuristic đơn giản. Mặc dù cách tiếp cận này giúp giảm thời gian xử lý, nhưng nó có thể không mang lại lựa chọn tốt nhất, dẫn đến missing cost cao hơn mức cần thiết. Do đó, một hướng nghiên cứu tiếp theo là tìm kiếm và áp dụng một phương pháp hiệu quả hơn để xác định item mục tiêu. Có thể xem xét sử dụng các thuật toán tối ưu hóa ngẫu nhiên hoặc các kỹ thuật tiên tiến khác để cải thiện độ chính xác trong việc lựa chọn item cần xóa, từ đó giảm thiểu tác động tiêu cực của quá trình ẩn dữ liệu.

Những hướng phát triển này không chỉ giúp hoàn thiện thuật toán SO2DI mà còn góp phần nâng cao hiệu quả của các phương pháp bảo vệ tính riêng tư trong khai thác mẫu hữu ích, mở ra triển vọng ứng dụng rộng rãi hơn trong thực tế.

Tài liệu tham khảo

Tiếng Anh

- [1] Nguyen, D., Tran, M.-T., and Le, B., “A new algorithm using integer programming relaxation for privacy-preserving in utility mining,” *Applied Intelligence*, vol. 2023, Article ID 10489-023-04913-w, 2023.
- [2] Lin, J. C.-W., Wu, T.-Y., Fournier-Viger, P., Lin, G., Zhan, J., and Voznak, M., “Fast algorithms for hiding sensitive high-utility itemsets in privacy-preserving utility mining,” *Eng. Appl. Artif. Intell.*, vol. 55, pp. 269–284, 2016.
- [3] Lin, J.-W., Hong, T.-P., Fournier-Viger, P., Liu, Q., Wong, J.-W., and Zhan, J., “Efficient hiding of confidential high-utility itemsets with minimal side effects,” *J. Exp. Theor. Artif. Intell.*, vol. 132, p. 103 360, 2017.
- [4] Fournier-Viger, P., Lin, J. C.-W., Chi, T.-T., and Nkambou, R., “A survey of high utility itemset mining,” *High-Utility Pattern Mining: Theory, Algorithms and Applications*, pp. 1–45, 2019.
- [5] Liu, Y., Liao, W., and Choudhary, A., “A two-phase algorithm for fast discovery of high utility itemsets,” in *Proc. 9th Pacific-Asia Conf. Adv. in Knowledge Discovery and Data Mining*, 2005, pp. 689–695.
- [6] Tseng, S. V., Wu, C. W., Shie, B. E., and Yu, P. S., “Up-growth: An efficient algorithm for high utility itemset mining,” in *Proc. 16th*

ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2010, pp. 253–262.

- [7] Tseng, V. S., Wu, C., Shie, B., and Yu, P. S., “Efficient algorithms for mining high utility itemsets from transactional databases,” *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1772–1786, 2013.
- [8] Liu, M. and Qu, J., “Mining high utility itemsets without candidate generation,” in *Proc. 21st ACM Int. Conf. on Information and Knowledge Management*, 2012, pp. 55–64.
- [9] Fournier-Viger, P., Wu, C., Zida, S., and Tseng, V. S., “Faster high utility itemset mining using estimated utility cooccurrence pruning,” in *Proc. 21st Int. Symp. on Methodologies for Intelligent Systems*, 2014, pp. 83–92.
- [10] Zida, S., Fournier-Viger, P., Lin, J. C.-W., Wu, C. W., and Tseng, V. S., “Efim: A highly efficient algorithm for high-utility itemset mining,” in *Proc. 14th Mexican Int. Conf. Artif. Intell.*, Springer, 2015, pp. 530–546.
- [11] Yeh, J.-S. and Hsu, P.-C., “Hhuif and msicf: Novel algorithms for privacy preserving utility mining,” *Expert Syst. Appl.*, vol. 37, no. 7, pp. 4779–4786, 2010.
- [12] Yun, U. and Kim, J., “A fast perturbation algorithm using tree structure for privacy preserving utility mining,” *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1149–1165, 2015.
- [13] Li, S., Nankun, M., Le, J., and Liao, X., “A novel algorithm for privacy preserving utility mining based on integer linear programming,” *Eng. Appl. Artif. Intell.*, vol. 81, pp. 300–312, 2019.
- [14] Nguyen, D. and Le, B., “A fast algorithm for privacy-preserving utility mining,” *J. Inf. Technol. Commun.*, vol. 2022, no. 1, pp. 12–22, 2022.

- [15] Lin, C.-W., Hong, T.-P., Wong, J.-W., Lan, G.-C., and Lin, W.-Y., “A ga-based approach to hide sensitive high utility itemsets,” *Sci. World J.*, vol. 2014, pp. 1–12, 2014.
- [16] Holland, J. H., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.
- [17] Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P., “Optimization by simulated annealing,” *science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [18] Dorigo, M., Maniezzo, V., and Colorni, A., “Ant system: Optimization by a colony of cooperating agents,” *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.
- [19] Kennedy, J. and Eberhart, R., “Particle swarm optimization,” in *Proc. Int. Conf. Neural Networks (ICNN)*, IEEE, vol. 4, 1995, pp. 1942–1948.
- [20] Geem, Z. W., Kim, J. H., and Loganathan, G. V., “A new heuristic optimization algorithm: Harmony search,” *simulation*, vol. 76, no. 2, pp. 60–68, 2001.
- [21] Mirjalili, S., Mirjalili, S. M., and Lewis, A., “Grey wolf optimizer,” *Advances in engineering software*, vol. 69, pp. 46–61, 2014.
- [22] Mirjalili, S., “The ant lion optimizer,” *Advances in engineering software*, vol. 83, pp. 80–98, 2015.
- [23] Dhiman, G. and Kumar, V., “Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications,” *Advances in Engineering Software*, vol. 114, pp. 48–70, 2017.
- [24] Kennedy, J. and Eberhart, R. C., “A discrete binary version of the particle swarm algorithm,” in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, IEEE, vol. 5, 1997, pp. 4104–4108.