|  |
| --- |
| **TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**  **TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**  **KHOA CÔNG NGHỆ THÔNG TIN**  **Description: logoTDT%20(NH%E1%BB%8E)**  **BÁO CÁO**  **DỰ ÁN CÔNG NGHỆ THÔNG TIN 2**  **ĐỀ TÀI: TÌM HIỂU VỀ ITEMSET MINING VÀ CÀI ĐẶT THUẬT TOÁN HIGH OCCUPANCY ITEMSET**  **GIÁO VIÊN HƯỚNG DẪN:** **DOÃN XUÂN THANH**  **TRẦN VĂN SÁNG - 51703173**  **TRẦN GIA THÁI - 51703184**      **THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024** |
| **TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**  **TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**  **KHOA CÔNG NGHỆ THÔNG TIN**  **Description: logoTDT%20(NH%E1%BB%8E)**  **BÁO CÁO**  **DỰ ÁN CÔNG NGHỆ THÔNG TIN 2**  **ĐỀ TÀI: TÌM HIỂU VỀ ITEMSET MINING VÀ CÀI ĐẶT THUẬT TOÁN HIGH OCCUPANCY ITEMSET**  **GIÁO VIÊN HƯỚNG DẪN: DOÃN XUÂN THANH**  **TRẦN VĂN SÁNG - 51703173**  **TRẦN GIA THÁI - 51703184**      **THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024** |

# LỜI CẢM ƠN

Dự án của chúng em đề cập đến việc tìm hiểu và triển khai thuật toán High Occupancy Itemset. Trong quá trình tìm hiểu về Itemset Mining và cài đặt thuật toán High Occupancy Itemset, chúng em gặp nhiều khó khăn do chưa có kinh nghiệm về đề tài. Chúng em xin gửi lời cảm ơn chân thành đến người hướng dẫn – thầy **Doãn Xuân Thanh** đã giúp đỡ và hỗ trợ chúng em trong quá trình thực hiện dự án.

Và nhóm em xin gửi lời cảm ơn đến Trường Đại học Tôn Đức Thắng đã tạo điều kiện cho chúng tôi được học hỏi thêm thực tế từ người hướng dẫn tận tâm – thầy **Doãn Xuân Thanh**. Nhờ sự hướng dẫn và am hiểu chuyên môn của thầy, chúng em đã có thể hoàn thành dự án của mình.

Em xin chân thành cảm ơn!

*TP. Hồ Chí Minh, ngày … tháng … năm 2024*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Trần Văn Sáng*

*Trần Gia Thái*

# LỜI CAM ĐOAN

Tôi xin cam đoan đây là sản phẩm dự án của riêng chúng tôi và được hướng dẫn của GV Doãn Xuân Thanh. Các nội dung nghiên cứu, kết quả trong dự án này là trung thực.

*TP. Hồ Chí Minh, ngày … tháng … năm 2024*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Trần Văn Sáng*

*Trần Gia Thái*

# PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

*TP. Hồ Chí Minh, ngày … tháng … năm 2024*

*(ký tên và ghi rõ họ tên)*

**Phần đánh giá của GV chấm bài**

*TP. Hồ Chí Minh, ngày … tháng … năm 2024*

*(ký tên và ghi rõ họ tên)*

**MỤC LỤC**

[LỜI CẢM ƠN 3](#_Toc161303161)

[LỜI CAM ĐOAN 4](#_Toc161303162)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 5](#_Toc161303163)

[DANH MỤC CÁC TỪ VIẾT TẮT 7](#_Toc161303164)

[DANH MỤC CÁC HÌNH VẼ 8](#_Toc161303165)

[PHẦN 1 - GIỚI THIỆU CHUNG ITEMSET MINING 9](#_Toc161303166)

[**1.1.** **Khái niệm** 9](#_Toc161303167)

[**1.2.** **Đặc điểm của Itemset Mining** 9](#_Toc161303168)

[**1.3.** **Các bước cơ bản thực hiện Itemset Mining** 9](#_Toc161303169)

[**1.4.** **Ứng dụng** 9](#_Toc161303170)

[PHẦN 2 - THUẬT TOÁN HEP 11](#_Toc161303171)

[**2.1.** **Khái niệm** 11](#_Toc161303172)

[**2.2.** **Thành phần cơ bản** 11](#_Toc161303173)

[**2.3.** **Ứng dụng** 11](#_Toc161303174)

[**2.4.** **Cách thức hoạt động** 11](#_Toc161303175)

[**2.5.** **Phân tích thuật toán:** 12](#_Toc161303176)

[PHẦN 3 - CẢI TIẾN THUẬT TOÁN HEP 17](#_Toc161303177)

[**3.1.** **Vì sao cần cải tiến thuật toán** 17](#_Toc161303178)

[**3.2.** **Phương pháp cải tiến** 17](#_Toc161303179)

[**3.3.** **Phân tích thuật toán FHOI** 18](#_Toc161303180)

[**3.4.** **Phân tích Thuật toán DFHOI** 18](#_Toc161303181)

[PHẦN 4 - PHÂN TÍCH CODE THỰC HIỆN THUẬT TOÁN. (PHẦN NÀY CỦA THÁI TRẦN NẾU CÓ) 20](#_Toc161303182)

[PHẦN 5 - DEMO 21](#_Toc161303183)

[**5.1.** **Vì sao cần cải tiến thuật toán** 21](#_Toc161303184)

[**5.2.** **Ý tưởng** 21](#_Toc161303185)

[**5.3.** **Các bước thực hiện** 21](#_Toc161303186)

[TÀI LIỆU THAM KHẢO 32](#_Toc161303187)

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| HEP | High Efficiency Pruning |
| FHOI | Fast High Occupancy Itemset Mining |
| DFHOI | Depth First Search for High Occupancy Itemset Mining |
| StSet | Support Transaction Set |
| UBO | Upper-bound Occupancy |
|  |  |
|  |  |
|  |  |

# DANH MỤC CÁC HÌNH VẼ

[Hình 5.1 Trang chủ Website 21](#_Toc161303188)

[Hình 5.2 Trang giỏ hàng 22](#_Toc161303189)

[Hình 5.3 Trang menu sản phẩm hàng bán 22](#_Toc161303190)

[Hình 5.4 Câu SQL 23](#_Toc161303191)

[Hình 5.5 File CSV 23](#_Toc161303192)

[Hình 5.6 Đọc dữ liệu từ file CSV 24](#_Toc161303193)

[Hình 5.7 In dữ liệu từ file CSV đã được đọc 24](#_Toc161303194)

[Hình 5.8 Chuyển đổi dữ liệu và tính toán 25](#_Toc161303195)

[Hình 5.9 Đọc dữ liệu danh sách có mục duy nhất 25](#_Toc161303196)

[Hình 5.10 Lấy ra danh sách có mục duy nhất 26](#_Toc161303197)

[Hình 5.11 Viết hàm đếm số Tid chứa vật phẩm duy nhất 26](#_Toc161303198)

[Hình 5.12 Viết hàm tính occupancy 27](#_Toc161303199)

[Hình 5.13 Viết hàm tính tỷ lệ sử dụng giới hạn trên 27](#_Toc161303200)

[Hình 5.14 Viết các hàm xây dựng occupancy list 28](#_Toc161303201)

[Hình 5.15 Viết các hàm xây dựng occupancy list 28](#_Toc161303202)

[Hình 5.16 Tạo dataframe hợp Occupancy, Support, UBO 29](#_Toc161303203)

[Hình 5.17 Viết hàm main và hàm test 29](#_Toc161303204)

[Hình 5.18 Viết hàm main và hàm test (tiếp theo) 30](#_Toc161303205)

[Hình 5.19 In ra kết quả và tiến hành export kết quả ra file CSV 30](#_Toc161303206)

[Hình 5.20 Import kết quả vào DB và hiển thị khi cần thiết 31](#_Toc161303207)

# GIỚI THIỆU CHUNG ITEMSET MINING

* 1. **Khái niệm**

Itemset Mining (khai thác tập hợp) là quá trình phân tích dữ liệu để tìm kiếm các tập hợp (itemset) của các item xuất hiện trong database. Itemset Mining cung cấp các phương pháp khai thác tập hợp dựa trên các chỉ số để xác định xem itemset đó có được xem là đặc biệt cho so với các itemset khác hay không. Một itemset có thể chứa 1 hoặc nhiều item dựa trên các phép kết hợp tồn tại trong database.

Itemset Mining có thể áp dụng trong nhiều lĩnh vực cụ thể để khai thác hành vi của khách hàng dựa trên các transaction được lưu trữ. Trong bài này, chúng tôi tập trung triển khai thuật toán High Occupancy Itemset (HOI). Thuật toán High Occupancy Itemset có 2 biến thể gồm FHOI (Fast High Occupancy Itemset) và DFHOI (DFS for High Occupancy Itemset). Đây là thuật toán nhận đầu vào (threshold) là 1 ngưỡng chấp nhận do người dùng thuật toán cài đặt, dựa trên chỉ số Occupancy, Support và UBO (Upper-bound Occupancy) để xác định 1 tập hợp có được xem là chiếm dụng cao hay không.

* 1. **Đặc điểm của Itemset Mining**

Transaction: Trong khai thác dữ liệu, một transaction là một chuỗi các sự kiện xảy ra liên tục trong một khoảng thời gian nhất định. Ví dụ, một chuỗi sự kiện mua hàng online như "đăng nhập -> tìm kiếm sản phẩm -> thêm vào giỏ hàng -> thanh toán" có thể được xem là một transaction.

Tập hợp (Itemset): Một nhóm một hoặc nhiều item trong transaction. Ví dụ, trong một siêu thị, một tập hợp mục có thể bao gồm "sữa" và "bánh mì".

Threshold: ngưỡng chấp nhận do người dùng thuật toán cài đặt.

Support: Chỉ số cho biết tần suất xuất hiện của một tập hợp cụ thể trong toàn bộ tập dữ liệu. Ví dụ, nếu 100 trong tổng số 1000 giao dịch bán hàng bao gồm cả "sữa" và "bánh mì", thì tập hợp mục đó có mức hỗ trợ là 10%.

Frequent Itemsets: Những tập hợp mục xuất hiện cùng nhau trong tập dữ liệu với mức độ thường xuyên vượt qua một ngưỡng hỗ trợ tối thiểu đã được định trước.

Occupancy: độ chiếm dụng trong transaction. Chỉ số này xác định xem 1 itemset chiếm bao nhiêu trong 1 transaction có trong database. Ví dụ: trong một transaction có chiều dài 10 item, trong đó có item “sữa” và “bánh mì” (tập hợp gồm 2 item) => Occupancy = 2/10 = 1/5.

UBO (Upper-bound Occupancy): ngưỡng chiếm dụng trên của tập hợp trong transaction. Đây là chỉ số dùng để tối ưu thuật toán, từ đó thuật toán sẽ không cần phải duyệt toàn bộ các itemset có thể có trong database nhưng vẫn giữ được hiệu quả do trả về các kết quả phù hợp với ngưỡng chấp nhận.

Occupancy-list: cấu trúc dữ liệu được áp dụng trong thuật toán HOI. Cấu trúc dữ liệu này gồm có 2 phần: item đang xét, transaction chứa item đó và chiều dài tương ứng của transaction. Ví dụ: xét item A, ta có Occupancy-list của item A = {'A': [(T1, 3), (T2, 3), (T4, 2), (T6, 5)]}. Trong đó, T1, T2, T4, T6 là id của transaction, 3, 3, 2, 5 là chiều dài tương ứng của mỗi Tid.

StStet (Support Transaction Set): tương tự như Occupancy-list, nhưng StSet chỉ cần lưu trữ các Tid có chứa tập hợp đang xét thay vì chứa cả chiều dài của transaction đó. Từ đó khi thực hiện thuật toán sẽ giúp giảm sử dụng dữ liệu, từ đó tăng tốc thuật toán. Đây là cấu trúc dữ liệu chính khi áp dụng thuật HOI. Ví dụ: xét item A, ta có StSet của item A = {'A': [T1, T2,T4,T6]}. Trong đó, T1, T2, T4, T6 là id của transaction.

High occupancy itemset: Một itemset được gọi là High Occupancy Itemset khi và chỉ khi tập đó có các chỉ số Occupancy, Support, UBO phù hợp với ngưỡng chấp nhận được đặt ra.

* 1. **Các bước cơ bản thực hiện Itemset Mining**

Xác định ngưỡng hỗ trợ tối thiểu (threshold): Đây là bước đầu tiên, quyết định một tập hợp được xem là Frequent Itemset hoặc trong bài báo cáo này là High Occupancy Itemset hay không khi và chỉ khi vượt qua được ngưỡng này.

Tìm kiếm các tập hợp mục tần suất cao: Sử dụng các thuật toán như Apriori, FP-growth để tìm ra các tập hợp mục tần suất cao.

Tìm kiếm các tập hợp mục có chiếm dụng cao: sử dụng thuật toán HOI đang được đề cập trong bài.

Phân tích và ứng dụng kết quả: Các tập hợp mục tần suất cao hoặc chiếm dụng cao có thể được sử dụng để phân tích mối quan hệ giữa các mục, giúp trong việc đưa ra quyết định về quản lý hàng tồn kho, tiếp thị, và khuyến mãi ….

* 1. **Ứng dụng**

Phân tích giỏ hàng: Giúp cửa hàng tối ưu hóa việc sắp xếp sản phẩm và tăng doanh số bán hàng bằng cách tìm hiểu nhóm sản phẩm mà khách hàng thường mua cùng nhau.

Gợi ý sản phẩm: Tạo ra các gợi ý sản phẩm dựa trên những gì khách hàng đã chọn, nhằm tăng cơ hội bán hàng chéo.

Phát hiện gian lận: Phân tích các mẫu giao dịch bất thường để phát hiện gian lận.

Như vậy, Itemset mining cung cấp các phương pháp mạnh mẽ cho việc phân tích dữ liệu và có thể áp dụng trong nhiều lĩnh vực khác nhau, từ bán lẻ đến ngân hàng và y tế, cung cấp thông tin chi tiết giúp cải thiện quyết định kinh doanh và chiến lược tiếp thị.

# THUẬT TOÁN HEP

* 1. **Khái niệm**

Thuật toán HEP(mining High Occupancy Itemset with Efficient Pruning strategy) là một trong những thuật toán nền tảng trong việc khai thác tập hợp chiếm dụng cao trong database. Thuật toán sử dụng chỉ số Occupancy để xác định 1 tập hợp có phải là high occupancy itemset khi và chỉ khi chỉ số Occupancy của tập đó lớn hơn hoặc bằng ngưỡng chấp nhận.

Trong quá trình khai thác tập chiếm dụng cao, không gian tìm kiếm trở nên rất lớn do mỗi tập hợp đều có thể là tập hợp chiếm dụng cao. Như vậy nếu chỉ đơn thuần thực hiện kiểm tra toàn bộ các tập hợp có thể có sẽ gặp vấn đề về bộ nhớ và xử lý. HEP ra đời nhằm mục đích tối ưu không gian tìm kiếm bằng cách sử dụng thêm chỉ số UBO (Upper-bound Occupancy). Như đã đề cập ở phần trên, chỉ số này giúp xác định từ đầu xem tập hợp đó có thỏa mãn ngưỡng chấp nhận, nếu thỏa mãn ngưỡng chấp nhận mới tiếp tục xét đến chỉ số Occupancy của tập hợp.

* 1. **Thành phần cơ bản của HEP**

- Đầu vào của thuật toán HEP: dữ liệu đầu vào để thuật toán hoạt động cần 2 cột: Tid (chứa các id của transaction), Items (danh sách các item xuất hiện trong Tid tương ứng, lưu ý các item có trong danh sách là các item duy nhất).

A screenshot of a black screen

Description automatically generated

- Occupancy: độ chiếm dụng của 1 itemset trong transaction được xác định dựa trên công thức sau:

A black and white symbol

Description automatically generated

Trong đó:

P: đại diện cho itemset đang xét.

O(P): chỉ số Occupancy của itemset đang xét.

- StSet(P): Support transaction Set của itemset đang xét.

|P| và |T|: chiều dài của tập itemset đang xét và chiều dài của transaction có chứa itemset đó.

Công thức trên định nghĩa các tính Occupancy cho itemset. Ví dụ: sử dụng data mẫu đã cung cấp, xét itemset {a}, ta có itemset {a} tồn tại trong các Tid T1, T2, T4, T6.



- UBO (Upper-bound Occupancy): ngưỡng trên Occupancy của itemset được xác định dựa trên công thức sau:



Trong đó:

n: số lượng chiều dài phân biệt của các transaction chứa itemset đang xét. Số lượng này được sắp xếp từ nhỏ đến lớn dựa trên chiều dài của transaction.

l: chiều dài phân biệt của transaction, được sắp xếp từ nhỏ đến lớn.

UBO của 1 itemset được xác định dựa trên UBO tối đa mà itemset đó có thể có. Ví dụ: sử dụng data mẫu đã cung cấp, xét itemset {a} tồn tại trong các Tid T1, T2, T4, T6. Như vậy chiều dài phân biệt của transaction chứa a bằng l({a}) = {2,3,5}; số lượng chiều dài phân biệt của transaction chứa a bằng n({a}) = {1,2,1}.

A math equations and symbols

Description automatically generated with medium confidence

=> Sử dụng các chỉ số UBO và Occupancy sẽ tính toán được itemset đó có được gọi là high occupancy itemset hay không nếu các 2 chỉ số này của itemset đều không nhỏ hơn threshold.

# CẢI TIẾN THUẬT TOÁN HEP

* 1. **Vấn đề của thuật toán HEP**

Thuật toán HEP cung cấp một phương thức mới để khai thác các itemset so với việc thực hiện khai khác frequent itemset. Thuật toán HEP cho ra kết quả các itemset được gọi là chiếm dụng cao có số lượng ít hơn nhiều khi so với frequent itemset. Vì số lượng kết quả nhỏ hơn nhiều, các nhà phân tích có thể tập trung vào các itemset để đi sâu phân tích sự liên quan giữa các item trong itemset với nhau.

Tuy nhiên, thuật toán HEP lại phải thực hiện tạo rất nhiều các itemset ứng viên, từ đó xác định các chỉ số thỏa mãn thuật toán rồi mới ra được kết quả. Quá trình này diễn ra rất lâu, cá nhân tôi khi thực hiện cài đặt thuật toán này trong một số dataset cũng thường xuyên phải chờ đợi một khoảng thời gian rất dài, điều này làm mất rất nhiều thời gian. Trong thực tế, nếu muốn triển khai thuật toán này đòi hỏi một cỗ máy có hiệu năng xử lý rất cao thì mới ra được kết quả. Dù thuật toán HEP cung cấp một chỉ số UBO để cắt tỉa đi các itemset không đạt yêu cầu, nhưng nhìn chung việc tính toán UBO cũng gây ra rất nhiều khó khăn vì quy trình này phức tạp, nhiều vòng lặp và cũng cần phải tính toán toàn bộ UBO cho các itemset ứng viên. Trong trường hợp nếu toàn bộ transaction của dataset cần tính toán độ chiếm dụng có chiều dài bằng nhau, việc tính toán UBO cho dataset này không khả thi do tốn quá nhiều thời gian.

=> Như vậy, các vấn đề khiến thuật toán HEP trở nên chậm chạp là khi thuật toán này yêu cầu quét toàn bộ dataset để tạo các itemset ứng viên và từ đây tiếp tục tính toán UBO cho các itemset ứng viên này.

* 1. **Phương pháp cải tiến**

Dựa trên các vấn đề được đề cập ở trên, thuật toán FHOI và DFHOI ra đời nhằm khắc phục các vấn đề của thuật toán HEP, từ đó tăng tốc quá trình tính toán để tạo ra các itemset có độ chiếm dụng cao. Từ thực nghiệm cá nhân, tôi nhận ra rằng hai thuật toán này đều đạt hiệu suất tốt hơn nhiều so với HEP. Ở Phần 4 của bài báo cáo, tôi có thực hiện cùng chạy một tập dataset trên cả ba thuật toán HEP, FHOI và DFHOI, thuật toán FHOI và DFHOI đều tỏ ra vượt trội hơn so với HEP khi trả ra kết quả sớm nhất trong khi HEP vẫn đang tiếp tục tính toán.

Đóng góp chính của 2 thuật toán FHOI và DFHOI xuất phát từ việc loại bỏ các itemset ứng viên không có tiềm năng trở thành itemset chiếm dụng cao. Các chiến lược để loại bỏ itemset ứng viên và cải thiện không gian tìm kiếm như sau:

- Thay vì sử Occupancy-list làm cấu trúc dữ liệu chính để thực hiện thuật toán, FHOI và DFHOI yêu cầu sử dụng StSet để làm giảm bộ nhớ lưu trữ.

- Sau khi tính toán tạo ra các itemset ứng viên có một phần tử (sau đây gọi là 1-itemset), gọi k là số lượng item có trong itemset ứng viên (sau đây gọi là k-itemset). Khi k >= 2, dựa trên StSet được tạo ra sau khi có itemset ứng viên, nếu chiều dài của StSet không lớn hơn hoặc bằng ngưỡng tối thiểu do người dùng định nghĩa, các itemset này sẽ bị loại bỏ trước khi tính toán UBO.

- Áp dụng tính chất lớp tương đương để tăng tốc quá trình tạo ứng viên. Trong quá trình thực hiện HEP, tôi nhận ra khi thuật toán khởi tạo các itemset ứng viên, các itemset ví dụ như {a,b} và {b,a} để được xét đến dù rằng chúng đều có vai trò tương đương trong tình huống tính toán itemset chiếm dụng cao. Việc áp dụng tính chất lớp tương đương sẽ rút ngắn được quá trình tạo ứng viên, các ứng viên như ví dụ đã trình bày sẽ không cần phải tính toán hai lần, từ đó đưa tốc độ tính toán nhìn chung của hai thuật toán tăng lên đáng kể.

- Việc áp dụng tính toán UBO là không cần thiết trong trường hợp tất cả các giao dịch trong cơ sở dữ liệu có cùng độ dài với nhau. Nhờ đó, khi FHOI và DFHOI sử dụng cho dataset mà các transaction có độ dài bằng nhau sẽ giúp tăng tốc hơn quá trình ra kết quả do không phải thực hiện tính toán UBO.

- Thuật toán DFHOI bắt nguồn từ FHOI. Tuy nhiên so với FHOI, bài nghiên cứu chỉ ra FHOI tiêu tốn nhiều bộ nhớ để khai thác tập itemset chiếm dụng cao. Thuật toán DFHOI áp dụng kĩ thuật để DFS bằng cách thực hiện đệ quy, từ đó sẽ không cần phải mang theo một lượng lớn data khi chạy thuật toán. Tuy nhiên với thực nghiệm của tôi, thuật toán FHOI và DFHOI hầu như cho thời gian không khác biệt đáng kể, thậm chí ngay trong Phần 4 trình bày chi tiết về cách cài đặt thuật toán và dữ liệu mẫu chúng tôi có, thuật toán DFHOI tỏ ra chậm chạp hơn nhiều khi so với FHOI dù cả hai cùng cung cấp số lượng itemset chiếm dụng cao như nhau.

* 1. **Phân tích thuật toán FHOI**

Thuật toán FHOI (Fast High Occupancy Itemset Mining) được giới thiệu là một phương pháp hiệu quả để khai thác các tập hợp mục có độ chiếm hữu cao dựa trên lớp tương đương và cắt tỉa sớm. Mục tiêu của thuật toán này là tối ưu hóa quá trình khai thác bằng cách giảm bớt số lượng ứng viên cần xem xét và cải thiện không gian tìm kiếm. FHOI sử dụng hai chiến lược chính để cắt tỉa các ứng viên và nâng cao hiệu suất:

Lập chỉ mục chiều dài giao dịch: Điều này giúp tiết kiệm bộ nhớ dành cho việc lưu trữ danh sách chiếm hữu (Occupancy-List - OL) bằng cách chỉ lưu ID giao dịch chứa mục và chiều dài của giao dịch đó.

Sử dụng tính chất của lớp tương đương: Áp dụng tính chất này giúp tăng tốc quá trình tạo ra ứng viên bằng cách loại bỏ nhanh các ứng viên không hợp lệ trước khi tính toán giới hạn trên của độ chiếm hữu.

Tóm lại, FHOI phát triển dựa trên việc khai thác các tập hợp mục có độ chiếm hữu cao (HOIs) một cách nhanh chóng và hiệu quả bằng cách giảm thiểu số lượng ứng viên cần xem xét và tối ưu hóa không gian tìm kiếm thông qua cắt tỉa sớm. Bên cạnh đó, thuật toán cũng giới thiệu một định lý mới và áp dụng nó cho một lớp cơ sở dữ liệu cụ thể, giúp loại bỏ bước tính toán giới hạn trên của độ chiếm hữu và do đó, tăng tốc quá trình khai thác cũng như giảm yêu cầu về bộ nhớ.

* 1. **Phân tích Thuật toán DFHOI**

Thuật toán DFHOI (Depth First Search for High Occupancy Itemset Mining) sử dụng kỹ thuật Tìm kiếm Theo Chiều Sâu (DFS) để khai thác tập hợp mục có độ chiếm hữu cao một cách hiệu quả và nhanh chóng, đồng thời giải phóng bộ nhớ không cần thiết ngay lập tức sau khi khai thác. DFHOI không cần phải duy trì một số lượng lớn ứng viên cùng một lúc, giúp giảm thiểu yêu cầu về bộ nhớ và tăng tốc độ xử lý.

**Đặc điểm:**

* Sử dụng DFS: DFS được áp dụng để khai thác tập hợp mục có độ chiếm hữu cao một cách đệ quy, giúp thuật toán nhanh chóng xác định được những tập hợp mục đủ điều kiện mà không cần xem xét quá nhiều ứng viên không cần thiết.
* Cắt tỉa sớm ứng viên: Dựa vào tính chất của lớp tương đương, DFHOI loại bỏ sớm các ứng viên không hợp lệ trước khi tính toán độ chiếm hữu trên cơ sở, giảm thiểu thời gian và tài nguyên máy tính cần thiết cho quá trình khai thác.
* Tối ưu hóa việc sử dụng bộ nhớ: Không giữ một số lượng lớn các ứng viên trong bộ nhớ, DFHOI giúp giảm thiểu đáng kể lượng bộ nhớ cần thiết cho quá trình khai thác.

**Quy trình hoạt động:**

* Khởi tạo: Xác định tập hợp mục 1-itemset và chỉ số độ dài giao dịch để tiết kiệm bộ nhớ cho việc lưu trữ danh sách chiếm hữu.
* Khai thác 1-itemsets: Sử dụng thông tin đã thu thập, DFHOI xác định những 1-itemsets có độ chiếm hữu cao, chuẩn bị cho việc khai thác các tập hợp mục có kích thước lớn hơn.
* Khai thác k-itemsets: Áp dụng DFS để đệ quy khai thác các tập hợp mục có kích thước lớn hơn từ các ứng viên đã xác định, loại bỏ những ứng viên không đủ điều kiện dựa vào độ chiếm hữu và lớp tương đương.

Kết quả của DFHOI cho thấy hiệu suất cao về thời gian xử lý và sử dụng bộ nhớ khi so sánh với thuật toán HEP và FHOI, đặc biệt trong các cơ sở dữ liệu dày đặc, nơi mỗi giao dịch chứa một số lượng lớn mục và tập hợp mục ứng viên có kích thước lớn. DFHOI là phương pháp hiệu quả và tối ưu cho việc khai thác tập.

# PHÂN TÍCH CODE THỰC HIỆN THUẬT TOÁN. (PHẦN NÀY CỦA THÁI TRẦN NẾU CÓ)

# DEMO

* 1. **Vì sao cần cải tiến thuật toán**

Nhiệm vụ khai thác tập hợp mục thường xuyên rất quan trọng trong việc khai thác quy tắc liên kết, do đó nhiều thuật toán đã được phát triển để giải quyết các vấn đề khai thác tập hợp mục thường xuyên. Apriori là một trong những phương pháp.

* 1. **Ý tưởng**

- Sử dụng thuật toán HEP để tính toán và đưa ra gợi ý cho khách hàng khi họ thêm sản phẩm vào giỏ hàng của mình. Trong đó các Itemset để thực hiện là dựa vào các đơn đặt hàng trước đây, từ đó sẽ tính được các sản phẩm thường xuyên được mua chung trong cùng một đơn hàng. Dựa vào đó sẽ lấy ra các sản phẩm gợi ý cho khách hàng khi họ thêm một sản phẩm vào giỏ hàng.

* 1. **Các bước thực hiện**

a) Tạo website bán hàng:

Tạo website đơn giản sử dụng framawork laravel gồm các chứ năng như hiện thị sản phẩm, thêm vào giỏ hàng, đặt hàng….

Deploy website bằng máy ảo homestead với tên miền shopcaygiong.com

Trang chủ website:

A screenshot of a computer

Description automatically generated

Hình 5.1 Trang chủ Website

Trang giỏ hàng;

A screenshot of a computer

Description automatically generated

Hình 5.2 Trang giỏ hàng

Tạo một slider hiện thị sản phẩm liên quan khi khách thêm sản phẩm vào giỏ hàng.

A screenshot of a computer

Description automatically generated

Hình 5.3 Trang menu sản phẩm hàng bán

b) Chuẩn bị dữ liệu để chạy thuật toán.

Chuẩn bị Item: sử dụng câu sql và lấy tất cả mã đơn hàng và tất cả mã sản phẩm thuộc đơn hàng đó sau đó xuất dữ liệu thành file csv.

A screenshot of a computer

Description automatically generated

Hình 5.4 Câu sql

A screenshot of a computer

Description automatically generated

Hình 5.5 File csv

c. Tiến hành chạy thuật toán với dữ liệu mẫu.

Đọc, in dữ liệu từ file csv và chuyển thành dạng dữ liệu mong muốn:

A screenshot of a computer

Description automatically generated

Hình 5.6 Đọc dữ liệu từ file csv

A screenshot of a computer

Description automatically generated

Hình 5.7 In dữ liệu từ file csv đã được đọc

Chuyển đổi dữ liệu và tính toán

A screenshot of a computer

Description automatically generated

Hình 5.8 Chuyển đổi dữ liệu và tính toán

Lấy ra danh sách có mục duy nhất.

A screenshot of a computer

Description automatically generated

Hình 5.9 Đọc dữ liệu danh sách có mục duy nhất

A screenshot of a computer

Description automatically generated

Hình 5.10 Lấy ra danh sách có mục duy nhất

Viết hàm đếm số Tid chứa vật phẩm duy nhất:

A screenshot of a computer

Description automatically generated

Hình 5.11 Viết hàm đếm số Tid chứa vật phẩm duy nhất

Viết hàm tính occupancy:

A screenshot of a computer

Description automatically generated

Hình 5.12 Viết hàm tính occupancy

Viết hàm tính tỷ lệ sử dụng giới hạn trên:

A screenshot of a computer

Description automatically generated

Hình 5.13 Viết hàm tính tỷ lệ sử dụng giới hạn trên

Viết các hàm xây dựng occupancy list:

A screenshot of a computer

Description automatically generated

Hình 5.14 Viết các hàm xây dựng occupancy list

A screenshot of a computer

Description automatically generated

Hình 5.15 Viết các hàm xây dựng occupancy list

Tạo dataframe hợp Occupancy, Support, UBO:

A screenshot of a computer

Description automatically generated

Hình 5.16 Tạo dataframe hợp Occupancy, Support, UBO

Viết hàm main và hàm test:

A screenshot of a computer

Description automatically generated

Hình 5.17 Viết hàm main và hàm test

A screenshot of a computer

Description automatically generated

Hình 5.18 Viết hàm main và hàm test (tiếp theo)

In ra kết quả và tiến hành export kết quả ra file csv:

A screenshot of a computer

Description automatically generated

Hình 5.19 In ra kết quả và tiến hành export kết quả ra file csv

Import kết quả vào DB và hiển thị khi cần thiết:

A screenshot of a computer

Description automatically generated

Hình 5.20 Import kết quả vào DB và hiển thị khi cần thiết

# TÀI LIỆU THAM KHẢO

FILE TẬP DỮ LIỆU (GÁNG LINK VÀO ĐÂY)