TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Ngô Chí Cường - 52100778**

**Lê Trần Phú – 52100572**

**ÁP DỤNG KHAI THÁC MẪU ĐỒ THỊ**

**NHẰM TỐI ƯU HÓA ĐỂ PHÂN TÍCH**

**VIỆC TỐI ƯU HÓA DOANH SỐ BÁN**

**HÀNG**

**DỰ ÁN CÔNG NGHỆ THÔNG TIN**

**KỸ THUẬT PHẦN MỀM**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Ngô Chí Cường - 52100778**

**Lê Trần Phú – 52100572**

**ÁP DỤNG KHAI THÁC MẪU ĐỒ THỊ**

**NHẰM TỐI ƯU HÓA ĐỂ PHÂN TÍCH**

**VIỆC TỐI ƯU HÓA DOANH SỐ BÁN**

**HÀNG**

**DỰ ÁN CÔNG NGHỆ THÔNG TIN**

**KỸ THUẬT PHẦN MỀM**

Người hướng dẫn

**ThS Dzoãn Xuân Thanh**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**LỜI CẢM ƠN**

Trước tiên, chúng em xin gửi tới thầy Dzoãn Xuân Thanh lời cảm ơn chân thành nhất. Thầy là người đã mang đến cho chúng em nhiều kiến thức, kinh nghiệm và trong lúc đồng hành để hoàn thành Dự Án Công Nghệ Thông Tin. Suốt quá trình đồng hành cùng chúng em, Thầy luôn đưa ra những kiến thức chuyên môn sâu rộng, đầy sự tận tâm và nhiệt huyết.

Chúng em cũng xin bày tỏ lòng biết ơn sâu sắc đến tất cả các giáo viên trong Khoa Công nghệ thông tin của Trường Đại học Tôn Đức Thắng. Nhờ những kiến thức và kinh nghiệm mà các thầy cô đã chia sẻ, chúng em đã có thể hoàn thiện bài báo cáo này với chất lượng tốt nhất.

Cuối cùng, chúng em xin gửi lời cảm ơn đến gia đình và các bạn trong nhóm thực hành của chúng em. Những người bạn này đã luôn ở bên chúng em trong suốt quá trình học tập và giúp đỡ chúng em, là nguồn động lực rất lớn giúp chúng em tiến bộ hơn trong môn học của mình.

Chúng em rất cảm kích và tự hào khi được học tập và trải nghiệm những điều tuyệt vời như vậy. Chúng em hy vọng rằng sẽ có thể tiếp tục nhận được sự giúp đỡ và hướng dẫn từ các thầy cô để phát triển bản thân và đóng góp vào xã hội. Một lần nữa, chúng em xin chân thành cảm ơn!

*TP. Hồ Chí Minh, ngày 19 tháng 7 năm 2024*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Ngô Chí Cường*

*Lê Trần Phú*

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

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

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

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

*TP. Hồ Chí Minh, ngày 19 tháng 07 năm 2024*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

*Ngô Chí Cường*

*Lê Trần Phú*

**USING GRAPH PATTERN MINING FOR ANALYZING OF**

**MAKING MAXIMUM PROFIT IN SELL CULTURE**

**TÓM TẮT**

Đề tài của chúng em sẽ trình bày về lý thuyết của việc Graph Mining dựa trên việc phân tích dữ liệu tĩnh không thay đổi theo thời gian và dữ liệu động được thay đổi theo thời gian và áp dụng các thuật toán xử lý dữ liệu thành các đồ thị con từ đó để phân tích như TKG,AER… Vừa củng cố các cơ sở lý thuyết đã nêu và đồng thời thể hiện tính ứng dụng của Graph Mining vào cuộc sống

**USING GRAPH PATTERN MINING FOR ANALYZING OF**

**MAKING MAXIMUM PROFIT IN SELL CULTURE**

**ABSTRACT**

Our topic will present the theory of Graph Mining based on the analysis of static data (which does not change over time) and dynamic data (which changes over time). We will apply data processing algorithms to form subgraphs for analysis, such as TKG,AER… . This will both reinforce the theoretical foundations that we will demonstrate and demonstrate the practical applications of Graph Mining in real life.

**MỤC LỤC**

[DANH MỤC HÌNH VẼ viii](#_Toc172719955)

[DANH MỤC BẢNG BIỂU x](#_Toc172719956)

[DANH MỤC CÁC CHỮ VIẾT TẮT xi](#_Toc172719957)

[CHƯƠNG 1. MỞ ĐẦU VÀ TỔNG QUAN ĐỀ TÀI 1](#_Toc172719958)

[1.1 Lý do chọn đề tài 1](#_Toc172719959)

[1.2 Mục tiêu thực hiện đề tài 1](#_Toc172719960)

[1.3 Đối tượng nghiên cứu 1](#_Toc172719961)

[1.4 Ý nghĩa đề tài 2](#_Toc172719962)

[CHƯƠNG 2. KHAI THÁC ĐỒ THỊ CON 3](#_Toc172719963)

[2.1 Mức độ quan trọng việc phát triển việc khai thác dữ liệu dưới dạng đồ thị và các ứng dụng của chúng. 3](#_Toc172719964)

[2.2 Khai thác đồ thị con 3](#_Toc172719965)

[2.2.1 Khái niệm 3](#_Toc172719966)

[2.2.2 Bài toán khai thác đồ thị con 4](#_Toc172719967)

[2.3 Thuật toán khai thác cơ bản đồ thị con (G-Span) 5](#_Toc172719968)

[2.3.1 Khái niệm đi kèm trong khai thác đồ thị con 5](#_Toc172719969)

[2.3.2 Thuật toán G-Span 6](#_Toc172719970)

[2.4 Thuật toán TKG 7](#_Toc172719971)

[2.4.1 Cơ sở lý thuyết thuật toán 7](#_Toc172719972)

[2.4.2 Ví dụ đi kèm 7](#_Toc172719973)

[2.4.3 Các cách tối ưu thuật toán đề xuất 8](#_Toc172719974)

[CHƯƠNG 3. KHAI THÁC MẪU ĐỒ THỊ THUỘC TÍNH ĐỘNG 11](#_Toc172719975)

[3.1 Đồ thị thuộc tính động 11](#_Toc172719976)

[3.1.1 Khái niệm đồ thị động 11](#_Toc172719977)

[3.1.2 Khái niệm đồ thị thuộc tính động 11](#_Toc172719978)

[3.1.3 Các quy tắc giới hạn khi khai thác đồ thị thuộc tính động 12](#_Toc172719979)

[3.2 Các khái niệm về khai thác đồ thị thuộc tính động 12](#_Toc172719980)

[3.2.1 Trend-Graph 12](#_Toc172719981)

[3.2.2 Quy luật khi viết về sự biến đổi của các node trong Graph 13](#_Toc172719982)

[3.2.3 Công thức tính độ tự tin của một mẫu cốt lõi so với dữ liệu đồ thị gốc 13](#_Toc172719983)

[3.2.4 Công thức tính độ dốc của một mẫu cốt lõi so với dữ liệu đồ thị gốc 13](#_Toc172719984)

[3.3 Thuật toán AER (Attribute Evolution Rules) 14](#_Toc172719985)

[3.3.1 Các bước thực hiện thuật toán AER 14](#_Toc172719986)

[3.3.2 .Cách tối ưu thuật toán AER 16](#_Toc172719987)

[CHƯƠNG 4. THIẾT KẾ VÀ HIỆN THỰC HỆ THỐNG 17](#_Toc172719988)

[4.1 Lời giới thiệu 17](#_Toc172719989)

[4.2 Các sơ đồ thiết kế hệ thống 17](#_Toc172719990)

[4.2.1 Sơ đồ hoạt động hệ thống 17](#_Toc172719991)

[4.2.2 Sơ đồ lớp hệ thống 18](#_Toc172719992)

[4.3 Thuật toán cốt lõi cho hệ thống 19](#_Toc172719993)

[4.3.1 Thuật toán TKG chung cho việc trả về Top-K đồ thị con phổ biến với kỹ thuật tối ưu Dynamic-Search và tiền xử lý thuật toán 19](#_Toc172719994)

[4.3.2 Thuật toán AER nhằm trả về Association Rules 21](#_Toc172719995)

[CHƯƠNG 5. KẾT QUẢ THỰC NGHIỆM 25](#_Toc172719996)

[5.1 Link Github dự án đi kèm 25](#_Toc172719997)

[5.2 Menu điều hướng 25](#_Toc172719998)

[5.3 Trang lấy các sản phẩm duy nhất 25](#_Toc172719999)

[5.4 Trang vẽ đồ thị 26](#_Toc172720000)

[5.4.1 Vẽ Top-K đồ thị con thường xuyên 26](#_Toc172720001)

[5.4.2 Vẽ đồ thị con thường xuyên với số lượng item 27](#_Toc172720002)

[5.4.3 Trang đề xuất sản phẩm thêm vào combo 28](#_Toc172720003)

[CHƯƠNG 6. KẾT LUẬN 32](#_Toc172720004)

[6.1 Kết luận 32](#_Toc172720005)

[6.1.1 Số liệu đo đạc về hiệu suất 32](#_Toc172720006)

[6.1.2 Dự đoán thời gian xử lý cho thuật toán 33](#_Toc172720007)

[6.1.3 Ưu điểm 33](#_Toc172720008)

[6.1.4 Khuyết điểm 33](#_Toc172720009)

[6.2 Hướng phát triển 34](#_Toc172720010)

[TÀI LIỆU THAM KHẢO 35](#_Toc172720011)

# DANH MỤC HÌNH VẼ

[Hình 2.3.1.3.1: Hình đại diện cho mã DFS của 1 đồ thị 6](#_Toc172595854)

[Hình 3.1.1.2: Hình đại diện cho một đồ thị động 11](#_Toc172595855)

[Hình 3.1.2.3: Hình đại diện cho một đồ thị thuộc tính động 11](#_Toc172595856)

[Hình 3.2.1.4: Hình ảnh thể hiện Trend-Graph 13](#_Toc172595857)

[Hình 4.2.1.5:Hình ảnh sơ đồ hoạt động cho hệ thống 18](#_Toc172595858)

[Hình 4.2.1.6: Hình ảnh thể hiện sơ đồ lớp cho hệ thống 18](#_Toc172595859)

[Hình 4.3.1.1.7: Hình ảnh mã giả cho thuật toán TKG 19](#_Toc172595860)

[Hình 4.3.2.1.8: Hình ảnh mã giả thuật toán AER-Miner 22](#_Toc172595861)

[Hình 5.1.9: Hình ảnh trang Navigation cho Hệ thống 25](#_Toc172595862)

[Hình 5.2.10: Hình ảnh khi mới mở trang Get Unique Items 25](#_Toc172595863)

[Hình 5.2.11: Hình ảnh khi nhập đủ dữ liệu và thực hiện thuật toán lên trang Get Unique Items 26](#_Toc172595864)

[Hình 5.3.1.12: Hình ảnh khi mới mở trang Top-K Frequent Subgraph 26](#_Toc172595865)

[Hình 5.3.1.13: Hình ảnh khi nhập đủ dữ liệu và thực hiện thuật toán cho trang Top-K Frequent Subgraph 27](#_Toc172595866)

[Hình 5.3.2.14: Hình ảnh khi mới mở trang getSubgraphByNumberItems 27](#_Toc172595867)

[Hình 5.3.2.15: Hình ảnh khi nhập đủ dữ liệu và thực hiện thuật toán cho trang getSubgraphByNumberItems 28](#_Toc172595868)

[Hình 5.3.3.16: Hình ảnh khi mới mở trang findBestRecommendation 28](#_Toc172595869)

[Hình 5.3.3.17: Hình ảnh khi up file lên và nhấn submit để ra dữ liệu chọn 29](#_Toc172595870)

[Hình 5.3.3.18: Hình ảnh khi chọn dữ liệu 29](#_Toc172595871)

[Hình 5.3.3.19: Hình ảnh khi nhấn Find Recommendation và có kết quả 30](#_Toc172595872)

[Hình 5.3.3.20: Hình ảnh khi nhấn Find Recommendation và không có kết quả 30](#_Toc172595873)

[Hình 6.1.1.21: Hình ảnh biểu đồ lý thuyết thể hiện mối quan hệ của dữ liệu số đồ thị xử lý và thời gain 31](#_Toc172595874)

[Hình 6.1.2.22: Hình ảnh biểu đồ được dự đoán cho hệ thống 32](#_Toc172595875)

# DANH MỤC BẢNG BIỂU

[Bảng 6.1.1.1: Hình ảnh về bảng thời gian sau khi thử nghiệm trên các kích thước dữ liệu khác nhau 31](#_Toc172595876)

# DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| AER | Attribute Evolution Rules |
| AI | Artificial Intelligence |
| BFS | Breadth First Search |
| CRM | Customer relationship management |
| CSS | Cascading Style Sheets |
| CSV | comma-separated values |
| DFS | Depth-first search |
| HIV | Human Immunodeficiency Viruses |
| HTML | Hypertext Markup Language |
| Sup | Support |
| TKG | Top-K frequent subgraph mining |

# MỞ ĐẦU VÀ TỔNG QUAN ĐỀ TÀI

## Lý do chọn đề tài

Hiện nay, AI đang là xu hướng của thế giới khi tất cả các ngành nghề nếu muốn được hiệu suất cao thì một trong những thứ tiên quyết nhất chính là việc áp dụng AI vào các trường hợp thực tế. Đặc biệt nhất trong môi trường kinh tế khi bài toán gia tăng hiệu suất kinh tế luôn là thứ tiên quyết được đặt lên hàng đầu do đó việc tận dụng AI vào kinh tế là một việc hết sức cần thiết đặc biệt trong thời đại hiện nay.

Với những niềm cảm hứng trên thì việc chọn 1 đề tài vừa áp dụng vào thực tế hiện nay nhất là trong lĩnh vực kinh doanh và mang tính học thuật về việc xử lý dữ liệu đã là động lực để chúng em hoàn thành đề tài về … nhằm hỗ trợ việc tạo ra các hệ thống hỗ trợ ra quyết định ở các cấp quản lý và với dự định xa hơn là phát triển hệ thống lên thành cấp chiến lược.

## Mục tiêu thực hiện đề tài

Với mục đích trên thì đề tài trên sẽ nhắm chủ yếu vào việc hỗ trợ người dùng ra quyết định dựa trên các dữ liệu có sẵn và ở đây chủ yếu là từ các giao dịch sản phẩm từ các cửa hàng vừa và nhỏ để từ đó có thể hỗ trợ tạo ra các gói ưu đãi nhằm từ 1 sản phẩm cho trước để tối ưu hóa doanh số bán hàng.

## Đối tượng nghiên cứu

Thuật toán nghiên cứu: Chủ yếu là từ các thuật toán của Graph Mining dựa trên việc phân tích dữ liệu tĩnh không thay đổi theo thời gian và dữ liệu động được thay đổi theo thời gian và áp dụng các thuật toán xử lý dữ liệu thành các đồ thị con từ đó phân tích như TKG,AER…

Dữ liệu nghiên cứu: Bộ dữ liệu hiển thị giao dịch của các cửa hàng bán lẻ vừa và nhỏ

Người sử dụng: Quản lý của các cửa hàng bán lẻ vừa và nhỏ

Công cụ nghiên cứu: Ngôn ngữ Python kèm với các framework hỗ trợ tạo ra Web dựa trên nền backend Python. Kèm với các template hỗ trợ render dưới dạng server như Jinja2 và các ngôn ngữ lập trình Web phổ biến như HTML, CSS và Javascript đi cùng các thư viện phổ biến như Jquery.

## Ý nghĩa đề tài

Việc triển khai thành công hệ thống sẽ mang ý nghĩa rất to lớn cho các chủ cửa hàng vừa và nhỏ do các ứng dụng hiện nay chưa phổ biến quá nhiều đặc biệt là các ứng dụng xử lý dữ liệu và học máy cho các chủ cửa hàng vừa và nhỏ do nhiều nguyên nhân đi kèm về vấn đề chi phí phát sinh và hơn nữa những đóng góp trên sẽ góp phần hỗ trợ để hình thành đóng góp lớn hơn và có nhiều sức ảnh hưởng hơn trong công cuộc chuyển đổi số ở Việt Nam hiện nay.

# KHAI THÁC ĐỒ THỊ CON

## Mức độ quan trọng việc phát triển việc khai thác dữ liệu dưới dạng đồ thị và các ứng dụng của chúng.

Biểu đồ càng ngày quan trọng trong cuộc sống hầu hết được xuất hiện trong các ngành liên quan đến khoa học như:

* Hóa học: Các công thức hợp chất hữu cơ ….
* Sinh học: Cấu trúc Protein, mạng sinh học
* Công nghệ thông tin: Mạng xã hội …
* Điện: Các mạch điện…

Nhờ đó các thuật toán tìm kiếm đồ thị đã được phát triển trong lĩnh vực thông tin hóa học, thị giác máy tính… từ đó đẩy mạnh quá trình phân tích dữ liệu có cấu trúc đồ thị trong thời gian gần đây.

Trong đó phân tích đồ thị con được xem là một trong những hoạt động được đầu tư nghiều nhất nhiều nhất vì sự hữu ích trong tập hợp đồ thị, phân biệt các nhóm đồ thị khác nhau, phân loại và gom cụm đồ thị, xây dựng chỉ số đồ thị và hỗ trợ tìm kiếm đồ thị tương tự trong cơ sở dữ liệu đồ thị…

Qua đó các loại thuật toán liên quan đến đồ thị đã được ra đời và đưa ra được những ứng dụng tuyệt vời như sau

* Y học: Phân tích dữ liệu cấu trúc HIV bằng việc so sánh giữa các đồ thị con thường xuyên với nhau
* Hóa học: Phân loại đặc trưng của cấu trúc các chất hóa học
* Kinh doanh: Phân tích các dữ liệu liên kết giữa các khách hàng và mặt hàng với nhau nhằm tối ưu hóa việc kinh doanh dựa trên việc phát triển CRM

## Khai thác đồ thị con

### Khái niệm

Một đồ thị sẽ được xem là đồ thị con của đồ thị còn lại nếu cả tập hợp tập đỉnh và tập cạnh của đồ thị đó là con của tập hợp các tập đỉnh và tập cạnh của các đồ thị còn lại

Ví dụ: Giả sử G là đồ thị với tập đỉnh VG ={1,2,3,4} và tập cạnh EG = {(1,2), (2,3), (3,4)} và H là đồ thị với tập đỉnh VG ={1,2,3} và tập cạnh EG = {(1,2), (2,3)}. Thì lúc này H sẽ là đồ thị con của đồ thị G.

### Bài toán khai thác đồ thị con

Đầu vào:

* Các Database là dạng đồ thị chứa các đối tượng đại diện là các node trong đồ thị và các cạnh nối các node nó
* Các giá trị ngưỡng (min-sup)

Đầu ra:

* Các đồ thị con mà đảm bảo được số lượng đồ thị con  Giá trị ngưỡng (min-sup)

#### Vấn đề đặt ra khi chọn ngưỡng trả về

Nếu ngưỡng quá cao 🡺 Quá ít kết quả trả về 🡺 Hạn chế tính ứng dụng

Ngưỡng quá thấp 🡺 Quá nhiều kết quả trả về 🡺 Hiệu suất tính toán giảm đáng kể và thời gian tính toán tăng đột biến

Điều kiện để chọn ngưỡng

* Tài nguyên lưu trữ và thời gian tính toán
* Độ phức tạp của các tập mẫu
* Tính chất của tập mẫu

Vấn đề này còn lớn hơn khi trong thực tế việc chọn ngưỡng cần phải đảm bảo tính ứng dụng ở một số khía cạnh như không thể trả về quá nhiều đồ thị con làm ảnh hưởng tới việc phân tích của người dùng và đồng thời bài toán hiệu suất cũng là thứ cần phải cân nhắc

#### Phương pháp giải quyết một phần

Chúng ta sẽ chuyển bài toán về dạng xếp hạng mô hình có độ phổ biến cao nhất theo top-k (top-k frequent subgraph mining) để giảm số lượng đồ thị con cần trả về với các đầu vào và đầu ra như sau:

* Đầu vào: số nguyên k, đại diện cho số lượng mô hình con cần được sinh ra cùng với giá trị ngưỡng (min-sup) tối thiểu để mô hình con đó được xem là phổ biến
* Đầu ra: Tập hợp gồm k mô hình con phổ biến nhất trong dữ liệu

Mặc dù vấn đề về số đồ thị con trả về trong thực tế đã được giải quyết nhưng những vấn đề cốt lõi của việc chọn giá trị ngưỡng vẫn chưa giải quyết khi chưa thể giải quyết hoàn toàn vấn đề hiệu suất cho từng mô hình nhằm tối ưu hóa về mặt thời gian lẫn độ chính xác và vẫn còn đang là một ẩn số rất khó để tìm được lời giải.

## Thuật toán khai thác cơ bản đồ thị con (G-Span)

### Khái niệm đi kèm trong khai thác đồ thị con

#### Định luật Aptori:

Nếu một đồ thị con đã không phổ biến thì tất cả các đồ thị chứa nó sẽ không phổ biến vì khi một đồ thị con được nhận ra không đủ phổ biến thì cho dù có như thế nào thì tất cả các đồ thị chứa đồ thị con đó cũng không được xem là phổ biến.

#### Hệ số hỗ trợ của đồ thị con:

Hệ số hỗ trợ của đồ thị con được định nghĩa là tần số xuất hiện của đồ thị con đó dưới đồ thị gốc.

#### Biểu diễn các cạnh dưới dạng DFS code:

A table with numbers and lines

Description automatically generated

Hình 2.3.1.3.1: Hình đại diện cho mã DFS của 1 đồ thị

Mỗi cạnh có thể được biểu diễn dưới dạng DFS code bằng công thức chung như sau (số thứ tự cạnh 1, Số thứ tự cạnh 2, tên cạnh 1, trọng số nối giữa cạnh 1 và 2, tên cạnh 2)

#### Cách nhận biết trùng lặp trong đồ thị con:

* Ta biết rằng mỗi đồ thị con tương tự như 1 đồ thị thì đều có thể biểu diễn dưới dạng DFS code(Search DFS cho đồ thị và ra kết quả)
* Khi chúng ta thêm 1 cạnh vào đồ thị con thì nếu DFS code của chúng không là duy nhất thì chúng ta nhận ra đồ thị con đó bị trùng lặp và đều đó dẫn đến không được thêm đồ thị con đó vào hàng đợi Qk để tránh tính chất duy nhất của đồ thị
* Kỹ thuật này được áp dụng vào G-Span để nhận ra trùng lặp

### Thuật toán G-Span

Cho 1 tập dữ liệu gồm các tập hợp các cạnh {A,B,C,D} và các trọng số {a,b,c,d} và các đồ thị cần xét.

Bước 1: Tiến hành nối các đỉnh với nhau bắt đầu bắt đầu từ đỉnh A-A với trọng số là a. Tìm hệ số hỗ trợ của đồ thị con trên đồ thị cần xét và thêm vào hàng đợi Qk.

Bước 2: Tiếp tục với A-B với trọng số là a và xét tương tự theo đỉnh rồi trọng số tới số lượng đỉnh của đồ thị con.

Kết hợp với cách nhận biết trùng lặp để tránh xét các đồ thị con đã xét xong và định luật Aptori nhằm giảm số lượng bước cần thực hiện và cho ra kết quả cuối cùng.

## Thuật toán TKG

### Cơ sở lý thuyết thuật toán

Bước 1: Khởi tạo với minsup (ngưỡng độ phổ biến tối thiểu) là 1

Bước 2: Bắt đầu dùng DFS để rã đồ thị thành các đồ thị con

Bước 3: Tạo một hàng đợi Qk chứa các đồ thị con và tần suất xuất hiện của chúng theo thứ tự từ lớn đến bé

Bước 4: Khi k đồ thị con được tìm thấy thì bắt đầu nâng min-sup của việc tìm kiếm lên bằng với hệ số hỗ trợ bé nhất của các phần tử trong Qk: khi đã tìm đc k đồ thị con thường gặp nhất, minsup sẽ được tăng lên bằng độ phổ biến của đồ thị con ít phổ biến nhất trong Qk, điều này giúp giảm không gian tìm kiếm và loại bỏ các đồ thị con có độ phổ biến thấp hơn.

Bước 5: Sau đó mỗi khi các phần tử được thêm vào thì nâng cao giá trị min-sup lên: với mỗi đồ thị con có mã DFS duy nhất được thêm vào, giá trị minsup sẽ được tăng lên tương ứng bằng với của đồ thị con ít phổ biến nhất trong Qk, điều này giúp tối ưu hóa quá trình tìm kiếm bằng việc loại bỏ các đồ thị con không cần thiết. Khi thuật toán kết thúc, top-k các đồ thị con phổ biến nhất đã được tìm thấy.

### Ví dụ đi kèm

Bài toán: Database có 3 đồ thị cơ bản: G1 (A – B – C), G2 (A – B, B – C, C – D), G3 (A – B – C – D). Áp dụng TKG như sau:

Bước 1: Khởi tạo minsup = 1: Ban đầu, minsup được đặt là 1. Thuật toán bắt đầu tìm kiếm các cạnh đơn trong các đồ thị.

Bước 2: Tìm kiếm từ các cạnh đơn: Các cạnh đơn được xác định là: (A-B), (B-C), và (C-D). Các cạnh này xuất hiện trong tất cả các đồ thị.

Bước 3: Tạo queue Qk: queue Qk sẽ chứa các đồ thị con phổ biến nhất. Ban đầu, các đồ thị cạnh con có các cạnh đơn trên được thêm vào queue này.

Bước 4: Điều chỉnh minsup: Khi đã tìm thấy đủ k mẫu (giả sử k=3), minsup được điều chỉnh dựa trên độ phổ biến của tiểu đồ thị ít phổ biến nhất trong Qk. Giả sử tiểu đồ thị (C-D) ít phổ biến nhất, minsup sẽ được tăng lên tương ứng.

Bước 5: Mở rộng các đồ thị con: Thuật toán tiếp tục mở rộng các đồ thị con bằng cách thêm các cạnh. Ví dụ, từ (A-B) có thể mở rộng thành (A-B-C).

Bước 6: Kiểm tra kết quả DFS: Mỗi khi một đồ thị con mới được thêm vào , kết quả DFS của nó được kiểm tra để đảm bảo không bị trùng lặp. Nếu trùng lặp, tiểu đồ thị đó bị bỏ qua.

Cụ thể: Khi mở rộng (A-B) thành (A-B-C)

* Đồ thị con (A-B-C) có giá trị support > minsup. Do đó, nó được coi là đồ thị con trong top-k phổ biến và có được thêm vào Qk.
* Sau đó, minsup có thể được tăng lên dựa trên số lượng xuất hiện của các đồ thị con khác trong Qk.

#### 🡺 Vấn đề đặt ra: Việc thêm vào tính giá trị support của đồ thị làm cho bài toán tối ưu không gian tìm kiếm là một vấn đề rất nan giải

### Các cách tối ưu thuật toán đề xuất

#### Kĩ thuật Dynamic-Search(Tìm kiếm động):

Tổng quan: Dynamic search là một trong những kỹ thuật được sử dụng để tối ưu hóa thuật toán TKG nhằm cải thiện hiệu suất bằng cách giảm không gian tìm kiếm. Phương pháp này tập trung vào việc tìm kiếm các đồ thị con có tần suất cao trước để nhanh chóng nâng ngưỡng tần suất tối thiểu (minsup), từ đó loại bỏ các đồ thị không tiềm năng

Quy trình:

* Khởi tạo
  + Giá trị minsup = 1
  + Khởi tạo priority queue Qk: lưu trữ các subgraph top-k phổ biến tại bất kỳ thời điểm nào. Các subgraph nào mà có giá trị support thấp sẽ có ưu tiên cao
  + Khởi tạo priority queue Qc: lưu trữ các subgraph mở rộng tiếp theo. Các subgraph nào mà có giá trị support cao sẽ có độ ưu tiên cao. Ban đầu queue này sẽ mang giá trị trống
* Các bước chạy thuật toán tối ưu
  + Trong khi Qc không rỗng:
* Lấy subgraph có độ ưu tiên cao nhất từ Qc
* Tìm tất cả các cạnh mở rộng của subgraph này và tính sup của chúng
* Đối với mỗi mở rộng của subgraph g, tạo subgraph g’ mới bằng cách thêm cạnh t vào DFS code của subgraph g ban đầu
* Nếu sup của g’ ≥ minsup và g’ là có mã DFS không bị trùng lặp, chèn g’ vào Qk
* Nếu kích thước của Qk > k, nâng ngưỡng minsup và loại bỏ đồ thị con có sup thấp nhất khỏi Qk.
* Chèn g’ vào Qc như một đồ thị con ưu tiên cho việc mở rộng trong tương lai

Kết thúc: trả về Qk chứa các đồ thị con phổ biến nhất tìm được.

#### Kĩ thuật loại bỏ đồ thị con không phổ biến

Bước 1: Cho đồ thị con g đã được lấy ra từ hàng đợi Qc, tìm kiếm và tính toán tần suất xuất hiện các đồ thị con và các đồ thị mở rộng của nó xuất hiện bên trong các đồ thị được cho sẵn.

Bước 2: Tìm tần suất cao nhất giữa các đồ thị đó với nhau gọi là hsup và rn là số lượng đồ thị còn lại không chứa bất kỳ đồ thị con g và mở rộng của chúng

Bước 3: Tính hsup + rn nếu tổng chúng bé hơn minsup thì chúng được xem là có đồ thị mở rộng không phổ biến và sẽ dừng bước tính toán đồ thị mở rộng ở các database còn lại.

#### Kĩ thuật tiền xử lý thuật toán

Tổng quan: Phương pháp trên quét cơ sở dữ liệu ban đầu nhằm tính toán sup của tất cả các đồ thị cạnh đơn trước khi thực hiện dynamic search, điều này giúp cập nhật các biến Qk, minsup và Qc, giảm thiểu thời gian xử lý cho các đồ thị cạnh đơn và tăng hiệu suất tổng thể

Quy trình:

* Scan: quét toàn bộ cơ sở dữ liệu để tính toán sup của tất cả các đồ thị cạnh đơn. Việc này nhằm xác định số lượng các lần xuất hiện của mỗi đồ thị cạnh đơn trong database
* Cập nhật Queue và minsup:
* Qk: thêm các đồ thị cạnh đơn có sup >= minsup vào Qk
* minsup: thiết lập minsup dựa trên giá trị hỗ trợ của các đồ thị cạnh đơn phổ biến nhất được tìm thấy
* Qc: thêm các đồ thị cạnh đơn có tiềm năng mở rộng vào Qc để tiếp tục mở rộng trong các bước tìm kiếm sau

Thực hiện dynamic search: sau khi cập nhật Qk, minsup, Qc, thực hiện dynamic search để khám phá các subgraph phổ biến hơn. Quy trình này bao gồm các bước mở rộng và kiểm tra các subgraph như đã mô tả trong phương pháp dynamic search

# KHAI THÁC MẪU ĐỒ THỊ THUỘC TÍNH ĐỘNG

## Đồ thị thuộc tính động

### Khái niệm đồ thị động

Một đồ thị mà bị thay đổi bởi yếu tố thời gian thì được gọi là một dynamic-graph

A diagram of a diagram

Description automatically generated

Hình 3.1.1.2: Hình đại diện cho một đồ thị động

### Khái niệm đồ thị thuộc tính động

Dynamic Attributed-Graph là một dynamic graph mà trong đó các node có thể được xem là 1 thực thể chứa các thuộc tính khác nhau

A diagram of a diagram

Description automatically generated

Hình 3.1.2.3: Hình đại diện cho một đồ thị thuộc tính động

### Các quy tắc giới hạn khi khai thác đồ thị thuộc tính động

Giới hạn về góc nhìn trong mô tả cho việc khai thác đồ thị thuộc tính động được nghiên cứu như sau

Giới hạn về pattern:

* Đồ thị tập trung vào các node có sự ảnh hưởng nhất định và họ hàng của chúng (significant trend sequences)
* Đồ thị mà các đỉnh có sự phát triển theo một quy luật như nhau và có thể không ảnh hưởng với nhau
* Đồ thị mà tập hợp các đỉnh không bị thay đổi theo thời gian

Giới hạn về quy luật:

* Chúng ta nên tập trung vào sự tiến hóa của tổng quan đồ thị khi khai thác thay vì để ý tới sự tiến hóa của các đỉnh bên trong chúng

## Các khái niệm về khai thác đồ thị thuộc tính động

### Trend-Graph

Đây là một đồ thị thể hiện sự biến thiên của các giá trị của các đồ thị gốc t1, t2… trong các khoảng thời gian ti1,ti2,… bằng các giá trị {+, -, =, ++, -- …}

A diagram of a mathematical equation

Description automatically generated

Hình 3.2.1.4: Hình ảnh thể hiện Trend-Graph

### Quy luật khi viết về sự biến đổi của các node trong Graph

Cách viết: (Subgraph thời gian t, Subgraph thời gian t+1, Giá trị đầu tiên, Giá trị tiếp theo)

### Công thức tính độ tự tin của một mẫu cốt lõi so với dữ liệu đồ thị gốc

Mục tiêu: Nhằm xác định xác suất xảy ra kết quả của node nguyên nhân



Trong đó:

V là đồ thị gốc cho trước

E là toàn bộ mẫu đơn cốt lõi

Sup là hệ số hỗ trợ so với đồ thị gốc

### Công thức tính độ dốc của một mẫu cốt lõi so với dữ liệu đồ thị gốc

Độ dốc thể hiện mức độ phụ thuộc các đỉnh Consequent so với các đỉnh Antecedent



Trong đó:

Conf là hệ số tự tin của mẫu cốt lõi đó so với đồ thị gốc

ExConf(antecedent(V,E)) là hệ số tự tin mong đợi của các mẫu đó so với đồ thị gốc

## Thuật toán AER (Attribute Evolution Rules)

Thuật toán AER (Attribute Evolution Rules) được đề xuất để khám phá một loạt đồ thị mẫu mới trong các đồ thị động có thuộc tính. Ý tưởng chung của thuật toán trên là từ việc tìm ra quy luật thay đổi các giá trị của Sub-Graph nhằm đưa ra về quy luật thay đổi của đồ thị chính

### Các bước thực hiện thuật toán AER

Bước 1: Biến đổi các giá trị liên tục thành các giá trị rời rạc

* Rời rạc hóa các giá trị liên tục: Chuyển các giá trị liên tục thành các nhóm rời rạc.

Ví dụ: Độ tuổi có thể được chuyển thành các nhóm tuổi: 0-12 (trẻ em), 13-19 (thiếu niên), 20-35 (thanh niên), 36-60 (trung niên), trên 60 (người cao tuổi).

* Gán nhãn cho các giá trị rời rạc: Mỗi nhóm giá trị rời rạc được gán một nhãn cụ thể.

Ví dụ: 0-12 được gán nhãn "trẻ em", 13-19 được gán nhãn "thiếu niên",…

Bước 2: Tìm kiếm các Consequent Node

* Xác định các Consequent Node: Tìm các đỉnh (nodes) trong đồ thị đại diện cho kết quả cuối cùng mà ta muốn phân tích.

Ví dụ: Trong một mạng xã hội, có thể chọn các đỉnh đại diện cho các sự kiện quan trọng như thay đổi trạng thái công việc, hôn nhân,…

* Tạo mẫu đơn: Mỗi mẫu đơn bao gồm một Consequent Node và các Antecedent Nodes (đỉnh nguyên nhân) liên kết với nó.

Ví dụ: Nếu Consequent Node là "thay đổi trạng thái công việc", thì Antecedent Nodes có thể là "thất nghiệp", "có việc mới",…

Bước 3: Mở rộng đồ thị

* **Mở rộng mẫu:** Bắt đầu mở rộng đồ thị theo kích thước k-1 (trong đó k là tổng số đỉnh Antecedent và Consequent) bằng cách thêm các đỉnh Antecedent vào đồ thị.

Bước 4: Tính lại các hệ số

* **Tính tần suất (support):** Đếm số lần mẫu xuất hiện trong tập dữ liệu.

Ví dụ: Mẫu {(a,+)} → {(b,−)} xuất hiện trong 10% của các đồ thị xu hướng.

* **Tính độ tin cậy (confidence):** Tính tỷ lệ giữa số lần xuất hiện của mẫu và số lần xuất hiện trong đồ thị xu hướng.

Ví dụ: Độ tin cậy của {(a,+)} → {(b,−)} là 70%, nghĩa là trong 70% của các trường hợp khi aaa tăng, bbb giảm.

* **Tính độ dốc (lift):** Đo lường mức độ mạnh mẽ của mối quan hệ giữa các đỉnh trong đồ thị xu hướng.

Ví dụ: Độ dốc của {(a,+)} → {(b,−)} cho thấy mức độ phụ thuộc của b vào a.

**Bước 5: Lọc các mẫu không phổ biến:**

Sử dụng các hệ số phổ biến để lọc các mẫu không đáng kể: Dựa trên định luật Aptori, các mẫu không thể phát triển được nữa sẽ bị loại bỏ.

Bước 6: Tiếp tục mở rộng

Tiếp tục mở rộng ở bước 3 với điều kiện tập hợp các mẫu có kích thước k-1 sẽ không có phần tử là rỗng hay đồng nghĩa với việc là các tập hợp còn lại phải đảm bảo việc có thể mở rộng ở toàn bộ các tập hợp nếu không thì sẽ chấm dứt thuật toán và trả về kết quả

### .Cách tối ưu thuật toán AER

* Định nghĩa chỉ số hỗ trợ và độ dốc phổ biến
* Khi mở rộng mẫu thì tuân theo BFS với độ ưu tiên theo bảng chữ cái nhằm tạo ra tính đồng nhất
* Giới hạn kích thước cho các mẫu
* Loại bỏ các phần tĩnh hoặc dao động không qua nhiều trong dữ liệu gốc

# THIẾT KẾ VÀ HIỆN THỰC HỆ THỐNG

## Lời giới thiệu

Hệ thống sẽ được thực hiện trên ngôn ngữ Python kèm với các framework hỗ trợ tạo ra Web dựa trên nền backend Python như Flask. Kèm với các template hỗ trợ render dưới dạng server như Jinja2 và các ngôn ngữ lập trình Web phổ biến như HTML, CSS và Javascript đi cùng các thư viện phổ biến như Jquery.

Trang Demo sẽ được chia ra làm 4 trang gồm 1 trang chính và 3 trang phụ mang tính chất trình bày khi 2 trang đầu sẽ tập trung vào hiện thực thuật toán TKG nhằm tìm ra được top-k mẫu sản phẩm được bán nhiều nhất và tìm ra top-3 sản phẩm mẫu với số lượng sản phẩm cụ thể. Trang tiếp theo sẽ là một trang phụ trợ để tìm ra các số lượng sản phẩm trong độc nhất và cuối cùng là một trang áp dụng thuật toán TKG và AER tạo ra một hệ thống hỗ trợ ra quyết định nên thêm sản phẩm gì tiếp theo khi tạo một gói combo dựa trên thông tin mua hàng của người dùng để mang lại doanh thu cao nhất.

## Các sơ đồ thiết kế hệ thống

Các sơ đồ thiết kế hệ thống sẽ tập trung vào tính năng cốt lõi nhất là hệ thống hỗ trợ ra quyết định thêm sản phẩm để mang lại cái nhìn tổng quan nhất.

### Sơ đồ hoạt động hệ thống

Sơ đồ hoạt động hệ thống trên dựa trên các thuật toán TKG và AER để thiết lập nhưng bao gồm cả việc nâng cao trải nghiệm người dùng khi nếu không tìm được sản phẩm dựa trên các chuỗi sản phẩm được đưa vào thì sẽ thực hiện hành động lọc ra sản phẩm cuối cùng để có thể tìm ra được sản phẩm có hệ số bán chạy dựa trên đó nhất để đưa về lời khuyên thêm sản phẩm nào vào.

A diagram of a diagram

Description automatically generated with medium confidence

Hình 4.2.1.5:Hình ảnh sơ đồ hoạt động cho hệ thống

### Sơ đồ lớp hệ thống

Sơ đồ lớp hệ thống đã cho thấy được các lớp findSubgraphWithNumberItems, findAllSuibgraph và findTopKSubgraph được dựa trên lớp TKG phát triển và phần chức năng finding\_best\_recommendation dựa trên lớp AER và AER dựa trên thuật toán TKG Mining để tiếp tục việc tính toán

A diagram of a computer

Description automatically generated

Hình 4.2.1.6: Hình ảnh thể hiện sơ đồ lớp cho hệ thống

## Thuật toán TKG chung cho việc trả về Top-K đồ thị con phổ biến với kỹ thuật tối ưu Dynamic-Search và tiền xử lý thuật toán

### Cơ sở mã giả TKG

1. input :Một đồ thị database, k: Số lượng đồ thị con cần trả về

2. output: the top-k frequent subgraphs

3. Tạo ra một hàng đợi ưu tiên Qk mà sẽ ưu tiên cho đồ thị con có hệ số hỗ trợ nhỏ hơn.

4. Tạo ra một hàng đợi Qc chứa đồ thị con mở rộng với việc ưu tiên hệ số hỗ trợ càng lớn.

5. Tạo ra một đồ thị rỗng g.

6. Đặt minsup = 1

7. while Qc không rỗng do

8. g ← đưa ra đồ thị con có hệ số hỗ trợ lớn nhất trong Qc

9. ε ← rightMostPathExtensions(g, GD) // Mở rộng đồ thị bằng cách thêm các cạnh từ đồ thị trên và tính hệ số hỗ trợ

10. foreach (t, sup(t)) ∈ ε do

11. g0 ← g ∪ {t} // Thêm cạnh vào DFS code của g và lưu đồ thị vào biến g0

12. sup(g0) ← sup(t) //Tính hệ số hỗ trợ cho g0

13. if sup(g0) ≥ minsup and isCanonical(g0) then

14. // Thêm g0 vào hàng đợi QK

16. if QK.size() ≥ k then

17. // Nâng giới hạn minsup

18. //Đưa ra khỏi hàng đợi Qk đồ thị có hệ số ưu tiên cao hơn (giá trị hỗ trợ thấp nhất) subgraph from QK

minsup = sup(QK.peek())

19. end

20. //Lưu g0 vào hàng đợi Qc cho mục đích mở rộng

21. Insert g0 into Qc

22. end

23. end

24. end

25. Return QK

Hình 4.3.1.1.7: Hình ảnh mã giả cho thuật toán TKG

### Các hàm hỗ trợ

#### Hàm hỗ trợ tìm RightMostPathExtension

1. def calculate\_support(subgraph,transactions):

2.     return sum(1 for transaction in transactions if subgraph.issubset(transaction))

3. def rightMostPathExtension(subgraph,transactions):

4.     EdgeExtension=[]

5.     subgraphList=[]

6.     transactionList=[]

7.     for node in subgraph:

8.         subgraphList.append(node)

9.     for transaction in transactions:

10.         transactionList=[]

11.         for item in transaction:

12.             transactionList.append(item)

13.         try:

14.             start\_index = transactionList.index(subgraphList[0])

15.             # Check if the subsequent elements match the entire subgraph

16.             if transactionList[start\_index:start\_index + len(subgraphList)] == subgraphList:

17.                 next\_item = transactionList[start\_index + len(subgraphList)] if start\_index + len(subgraphList) < len(transactionList) else None

18.                 foundNext=False

19.

20.                 if(next\_item!=None):

21.                     foundNext = any(item and item.get('Extension') == next\_item for item in EdgeExtension)

22.                 else:

23.                     foundNext=True

24.                 if not foundNext and next\_item!=None:

25.                     newSubgraph=frozenset(subgraph|{next\_item})

26.                     newSupport=calculate\_support(newSubgraph,transactions)

27.                     EdgeExtension.append({"Extension":next\_item,"Support":newSupport})

28.         except ValueError:

29.             continue

30.     return EdgeExtension

Đoạn code trên có thể giải thích đơn giản như sau

Theo lý thuyết RightMostPathExtension chúng ta có thể trả về đường đi trên bằng cách thêm 1 cạnh tiếp tục từ cạnh cuối cùng của đồ thị con và theo lý thuyết được ghi trên G-Span thì ta có thể giải thích đoạn code trên như sau

* Từ dòng 2-12 thì ta tiến hành biến các set Transaction và đồ thị con thành kiểu dữ liệu list nhằm thao tác index dễ dàng hơn
* Từ dòng 12 thì ta tiến hành lấy item đầu tiên của đồ thị con và tìm ra chỉ số index của đồ thị cha chứa item đó nếu có
* Tiến hành tính toán độ dài đồ thị con và lấy item từ chỉ số index đó tới chỉ số index đó + chiều dài đồ thị con và so sánh với đồ thị con
* Nếu đúng thì tiến hành lấy item đằng sau đồ thị con đó và ghép với đồ thị con và tính hệ số hỗ trợ.
* Dữ liệu trả về sẽ là dữ liệu các cạnh có thể mở rộng của đồ thị đó và hệ số hỗ trợ đi kèm

#### Hàm hỗ trợ kiểm tra xem DFS của đồ thị đó có phải nhỏ nhất hay không

1. class Graph:

2.     def \_\_init\_\_(self):

3.         self.adjacency\_list = {}

4.

5.     def add\_edge(self, start, end):

6.         if start not in self.adjacency\_list:

7.             self.adjacency\_list[start] = []

8.         if end not in self.adjacency\_list:

9.             self.adjacency\_list[end] = []

10.         self.adjacency\_list[start].append(end)

11.         self.adjacency\_list[end].append(start)

12.

13.     def get\_dfs\_code(self):

14.         # Start DFS from the lexicographically smallest node

15.         nodes = (self.adjacency\_list.keys())

16.         nodesList=[]

17.         for node in nodes:

18.             nodesList.append(node)

19.         if not nodes:

20.             return []

21.         start\_node = nodesList[0]

22.         visited = set()

23.         code = []

24.         self.\_dfs(start\_node, visited, code)

25.         return code

26.

27.     def \_dfs(self, current, visited, code):

28.         visited.add(current)

29.         for neighbor in (self.adjacency\_list[current]):

30.             if neighbor not in visited:

31.                 code.append({current, neighbor})

32.                 self.\_dfs(neighbor, visited, code)

33.

34. def is\_canonical(subgraph):

35.     newGraph=[]

36.     subgraphList=[]

37.

38.     for node in subgraph:

39.         subgraphList.append(node)

40.     for i in range (0,len(subgraphList)-1,1):

41.         newGraph.append({subgraphList[i],subgraphList[i+1]})

42.     graph = Graph()

43.     for start, end in newGraph:

44.         graph.add\_edge(start, end)

45.     g00 = graph.get\_dfs\_code()

46.     return newGraph == g00

47.

Thuật toán này có thể tóm tắt như sau:

* Từ dòng 34-41 thì ta tiến hành biến các set đồ thị con thành kiểu dữ liệu list nhằm thao tác index dễ dàng hơn và tạo ra DFS code cho các đồ thị
* Dòng 44 sẽ tiến hành khởi tạo đồ thị từ các DFS code được tạo ra bằng cách thêm các cạnh theo tuần tự và mô tả đồ thị dưới dạng các danh sách kề.
* Dòng 45 sẽ tiến hành DFS đồ thị để đưa ra được DFS nhỏ nhất của đò thị
* Dòng 46 tiến hành trả ra kết quả so sánh đồ thị với DFS code nhỏ nhất nhằm tìm xem đồ thị con có bị lặp lại không.

### Thuật toán hệ thống

1. def find\_top\_k\_subgraphs\_author(transactions, k, min\_support):

2.     Qk = []  # List to store top-k subgraphs

3.     Qc = []  # List for extending subgraphs

4.     # Function to calculate support of a subgraph

5.     def calculate\_support(subgraph):

6.         return sum(1 for transaction in transactions if subgraph.issubset(transaction))

7.

8.     for transaction in transactions:

9.         subgraphList=[]

10.         for node in transaction:

11.             subgraphList.append(node)

12.         if len(transaction) > 1:

13.             flagNotToDo=False

14.             subgraph = frozenset(subgraphList[:2])  # Chọn hai mục đầu tiên

15.             for \_,transanctionitem in Qk:

16.                 if subgraph.issubset(transanctionitem):

17.                     flagNotToDo=True

18.             if flagNotToDo ==True:

19.                 continue

20.             support = calculate\_support(subgraph)

21.             if support >= min\_support:

22.                 Qc.append((support, subgraph))

23.                 Qk.sort(key=lambda x: x[0])  # Sort Qk by support in ascending order

24.                 Qk.append((support, subgraph))

25.                 if len(Qk) >= k:

26.                     min\_support = Qk[0][0]  # Update minsup to the smallest support in Qk

27.                     if len(Qk) > k:

28.                         Qk.pop(0)  # Remove the subgraph with the least support if Qk exceeds size kss

29.         # Explore larger subgraphs based on the highest support in Qc

30.

31.     while Qc:

32.         Qc=sorted(Qc,reverse=True, key=lambda x: x[0])  # Sort Qc by support in descending order

33.         current\_support, current\_subgraph = Qc.pop(0)  # Get subgraph with highest support

34.         if current\_support > min\_support:

35.             current\_subgraph = set(current\_subgraph)

36.             all\_extension= rightMostPathExtension(current\_subgraph,transactions)

37.             for item in all\_extension:

38.                 edge = item.get("Extension")

39.                 support = item.get("Support")

40.                 new\_subgraph = frozenset(current\_subgraph | {edge})

41.                 if support >= min\_support and is\_canonical(new\_subgraph):

42.                     Qc.append((support, new\_subgraph))

43.                     Qk.append((support, new\_subgraph))

44.                     Qk=sorted(Qk,key=lambda x: x[0])

45.                     if len(Qk) >= k:

46.                         min\_support = Qk[0][0]  # Update minsup to the smallest support in Qk

47.                         if len(Qk) > k:

48.                               # Sort Qk by support in ascending order

49.                             Qk.pop(0)  # Remove the subgraph with the least support if Qk exceeds size k

50.     # Return top-k subgraphs with their support

51.     top\_k\_results = [(list(subgraph), support) for support, subgraph in Qk]

52.     return top\_k\_results

53.

Đoạn code trên có thể giải thích đơn giản như sau

* Dòng thứ 2-3 là những bước khởi tạo các giá trị Qk,Qc.
* Dòng 5-6 là hàm hỗ trợ đếm hệ số hỗ trợ Subgraph so với Transaction
* Từ dòng 9-28 là kỹ thuật tối ưu đầu tiên là tiền xử lý thuật toán khi tách và tính hệ số hỗ trợ các đồ thị con có 1 cạnh đơn và so sánh để loại bớt các cạnh có hệ số hỗ trợ bé hơn hệ số hỗ trợ tối thiểu theo định luật Aptori. Các đồ thị con đơn đạt đủ chỉ số và duy nhất sẽ bắt đầu thêm vào hàng đợi Qc
* Từ dòng 31-50 là kỹ thuật tối ưu tiếp theo là Dynamic Search khi bắt đầu mở rộng các cạnh từ các đồ thị con trên bằng các bước sau đây
  + Dòng 32-33: Sort các đồ thị con theo chỉ số support và lấy ra đồ thị đơn có chỉ số support cao nhất
  + Dòng 34-50:
    - Lấy phần tử đồ thị con đầu tiên của Qc và sau đó bắt đầu kiểm tra điều kiện hệ số hỗ trợ để quyết định có tiến hành mở rộng hay không
    - Bắt đầu tiến hành mở rộng đồ thị Subgraph bằng cách tìm các rightMostPathExtension của đồ thị con đó.
    - Nếu hệ số của Extension đó lớn hơn min-support và đồ thị đó có Caminomal nếu đáp ứng được thì tiến hành gắn nút đỏ vào nút cuối cùng của đồ thị con và tiến hành
    - Nếu đồ thị con có hệ số hỗ trợ cao hơn min\_support thì tiến hành thêm vào hàng đợi Qc để tiếp tục xét và mở rộng và thêm vào hàng đợi Qk.
    - Tiếp theo kiểm tra kích thước Qk nếu kích thước lớn hơn hoặc bằng k thì bắt đầu tiến hành cập nhật min-sup và loại bỏ đối tượng có min-sup thấp nhất
* Cuối cùng trả về hàng đợi Qk là hàng đợi chứa Top-K đồ thị phổ biến nhất.

### Ví dụ chi tiết

Bảng dữ liệu thuật toán TKG:

Cho minsup = 2 và tập dữ liệu Transaction sau

|  |
| --- |
| Transaction |
| a,b,c |
| a,b |
| a,d |
| a,b,d |
| b,c,d |
| a,c,d |

Bảng 4.3.1.1: Bảng thể hiện giá trị tập dữ liệu ví dụ thuật toán

* Từ dòng 10-19 thuật toán sẽ lấy ra các đồ thị đơn của dữ liệu trên và tính toán tần số của chúng của bảng Transaction và đưa vào hàng đợi Qc nếu hệ số hỗ trợ của chúng lớn hơn minsup
  + Đồ thị a-b: thuật toán tính ra hệ số hỗ trợ là 3 > minsup nên được thêm vào hàng đợi Qc,Qk
  + Đồ thị a-d: thuật toán tính ra hệ số hỗ trợ là 1 < minsup nên không được thêm vào hàng đợi Qc
  + Đồ thị b-c: thuật toán tính ra hệ số hỗ trợ là 2 = minsup nên được thêm vào hàng đợi Qc, Qk
  + Đồ thị a-c: thuật toán tính ra hệ số hỗ trợ là 1 < minsup nên không được thêm vào hàng đợi Qc
* Bắt đầu lấy giá trị đầu tiên có hệ số hỗ trợ cao nhất là đồ thị a-b ra khỏi hàng đợi Qc
  + Kiểm tra hệ số hỗ trợ đồ thị a-b là 3 > min-sup là 2 do đó ta thấy thỏa mãn và bắt đầu tiến hành mở rộng
  + Ta mở rộng đồ thị a-b bằng cách tìm cách tìm các đường mở rộng bằng hàm RightMostPathExtension:
    - Ta nhận thấy nếu duyệt a-b thì nút cuối cùng được duyệt sẽ là nút b nên ta sẽ tìm cạnh mở rộng dựa trên nút b.
    - Trong bảng dữ liệu Transaction thì chỉ có Transaction đầu tiên a-b-c thỏa mãn có đồ thị con a-b nên ta xem như cạnh bên phải của nút b là c trong Transaction do đó b-c là 1 cạnh có thể mở rộng.
    - Ta tính hệ số hỗ trợ của a-b-c là 1 < minsup nên không thỏa điều kiện và tiến hành khẳng định c không phải là nút có thể mở rộng từ nút cuối cùng trong danh sách duyệt đồ thị là b.
    - Do không còn extension nào nên tiến hành dừng vòng lặp
* Bắt đầu lấy giá trị đầu tiên có hệ số hỗ trợ cao nhất là đồ thị b-c ra khỏi hàng đợi Qc
  + Kiểm tra hệ số hỗ trợ đồ thị b-c là 2 = min-sup là 2 do đó ta thấy thỏa mãn và bắt đầu tiến hành mở rộng
  + Ta mở rộng đồ thị b-c-d bằng cách tìm cách tìm các đường mở rộng bằng hàm RightMostPathExtension:
    - Ta nhận thấy nếu duyệt b-c thì nút cuối cùng được duyệt sẽ là nút c nên ta sẽ tìm cạnh mở rộng dựa trên nút c.
    - Trong bảng dữ liệu Transaction thì chỉ có Transaction đầu tiên b-c-d thỏa mãn có đồ thị con a-b nên ta xem như cạnh bên phải của nút b là c trong Transaction do đó b-c là 1 cạnh có thể mở rộng.
    - Ta tính hệ số hỗ trợ của b-c-d là 1 < minsup nên không thỏa điều kiện và tiến hành khẳng định d không phải là nút có thể mở rộng từ nút cuối cùng trong danh sách duyệt đồ thị là b.
* Kết thúc thuật toán TopK đồ thị con thỏa điều kiện là a-b và a-c

## Thuật toán AER nhằm trả về Association Rules

### Cơ sở mã giả AER

1. input : Một đồ thị động hoặc một cái Trend-Graph, hệ số hỗ trợ tối thiểu, độ dốc tối thiểu and độ tự tin tối thiểu.

2. output: Toàn bộ các quy luật phù hợp

3. Khởi tạo một biến mẫu cốt lõi bằng kiểu dữ liệu Map với mapcandidates = <core pattern, instances> for cho việc phát triển patterns. Khởi tạo bằng dữ liệu ban đầu đồ thị.

4. Tạo ra một mảng chứa toàn bộ các mẫu trả về phù hợp với điều kiện.

5. While mapcandidates khác rỗng do

6. mapk+1 = ∅ (rỗng)

7. foreach candidate ∈ mapcandidates do

8. pattern ← candidate.key

9. instances ← candidate.value

10. foreach attr ∈ attributelist do

11. if attr ≥ thuộc tính cuối cùng của pattern và attr ≠ “=” then

12. newPattern ← pattern ∪ attr

13. newInstances ← extendInstances(pattern, attr, instaces)

14. support ⇐ sizeofnewInstance

15. lift, confidence ← calLiftAndConfi(pattern, attr, instaces)

16. if support ≥ minsup and lift ≥ minlift then

17. Đưa <newPattern,newInstance> ∈ mapk+1

18. end

19. end

20. end

21. end

22. listpatterns ← listpatterns ∪ map k+1

23. end

24. listpatterns ← filterPatterns(listpatterns, minconf)

25. Return AERs = listpatterns ∪ mergePatterns(listpatterns)

Hình 4.3.2.1.8: Hình ảnh mã giả thuật toán AER-Miner

### Thuật toán hệ thống

1. import itertools

2. def calculate\_support(pattern, frequencies):

3.     # Calculate the support of a pattern based on frequencies

4.     support = 0

5.     for items, freq in frequencies.items():

6.         if all(item in items for item in pattern):

7.             support += freq

8.     return support

9.

10. def calculate\_confidence(antecedent, consequent, frequencies):

11.     rule\_support = calculate\_support(antecedent + consequent, frequencies)

12.     antecedent\_support = calculate\_support(antecedent, frequencies)

13.     return rule\_support / antecedent\_support if antecedent\_support else 0

14.

15. def calculate\_lift(antecedent, consequent, frequencies):

16.     rule\_support = calculate\_support(antecedent + consequent, frequencies)

17.     antecedent\_support = calculate\_support(antecedent, frequencies)

18.     consequent\_support = calculate\_support(consequent, frequencies)

19.     total\_support = sum(frequencies.values())

20.     expected\_confidence = (antecedent\_support / total\_support) \* (consequent\_support / total\_support)

21.     return rule\_support / (expected\_confidence \* total\_support) if expected\_confidence else 0

22.

23. def AER\_Transaction\_Rules(transactions,minsup=0.03,minlift=0.04,minconf=0.02):

24.     # Extract items and their frequencies

25.     frequencies = {tuple(sublist): freq for sublist, freq in transactions}

26.     # Get unique items

27.     attribute\_list = sorted(set(itertools.chain(\*[list(item) for item in frequencies.keys()])))

28.

29.     # Initialize core patterns

30.     map\_candidates = [{tuple([attr]): frequencies} for attr in attribute\_list]

31.     # Step 2: Pattern Growth

32.     list\_patterns = []

33.     k = 1

34.

35.     while map\_candidates:

36.         new\_map\_candidates = []

37.         for candidate in map\_candidates:

38.             for pattern, instances in candidate.items():

39.                 for attr in attribute\_list:

40.                     if attr > pattern[-1]:

41.                         new\_pattern = pattern + (attr,)

42.                         new\_support = calculate\_support(new\_pattern, frequencies)

43.                         support = new\_support / sum(frequencies.values())

44.                         if support >= minsup:

45.                             confidence = calculate\_confidence(pattern, (attr,), frequencies)

46.                             lift = calculate\_lift(pattern, (attr,), frequencies)

47.                             if lift >= minlift:

48.                                 new\_map\_candidates.append({new\_pattern: frequencies})

49.                                 if confidence >= minconf:

50.                                     list\_patterns.append({

51.                                         'antecedent': list(pattern),

52.                                         'consequent': [attr],

53.                                         'support': support,

54.                                         'confidence': confidence,

55.                                         'lift': lift

56.                                     })

57.         map\_candidates = new\_map\_candidates

58.         k += 1

59.     return list\_patterns

Đoạn code trên có thể giải thích đơn giản như sau:

* Từ dòng 25-30 là các dòng hỗ trợ việc trích xuất ra đồ thị cốt lõi đơn từ các đồ thị con và tần số của chúng đặc biệt sắp xếp các thuộc tính đồ thị con theo bảng chữ cái
* Từ dòng 31-59 là cốt lõi thuật toán AER
  + Khởi tạo mảng gồm các mẫu có tiềm năng mở rộng với giá trị ban đầu là một mảng rỗng
  + Dòng 40 nhằm để so sánh việc có thể mở rộng được hay không bằng việc so sánh xem attribute có phải product đứng sau của pattern nếu có thì tiếp tục.
  + Dòng 41-46 nhằm tính lại các hệ số hỗ trợ đã được nêu công thức ở phía trên
  + Dòng 44,47,49 nhằm kiểm tra xem định luật có thỏa điều kiện phổ biến không
  + Các định luật phổ biến sẽ được cập nhật vào nhóm mảng có tiềm năng mở rộng tiếp tục
  + Tăng k lên và lặp lại bước trên

### Ví dụ thuật toán

Với tập dữ liệu ví dụ trên ta có thể chạy từng bước thuật toán trên như sau:

Với dữ liệu trên thì thuật toán sẽ lấy ra các

# KẾT QUẢ THỰC NGHIỆM

## Link Github dự án đi kèm

<https://github.com/OhJiKang/Final-Project-Technology/>

## Menu điều hướng

A screenshot of a computer

Description automatically generated

Hình 5.1.9: Hình ảnh trang Navigation cho Hệ thống

Menu trên sẽ điều hướng sang các trang còn lại

## Trang lấy các sản phẩm duy nhất

A screenshot of a computer

Description automatically generated

Hình 5.2.10: Hình ảnh khi mới mở trang Get Unique Items

Khi đưa file csv lên thì sẽ lấy được các sản phẩm độc nhất từ file csv đó

A screenshot of a computer

Description automatically generated

Hình 5.2.11: Hình ảnh khi nhập đủ dữ liệu và thực hiện thuật toán lên trang Get Unique Items

## Trang vẽ đồ thị

### Vẽ Top-K đồ thị con thường xuyên

Dựa trên thuật toán TKG đã được nêu ở phần trên thì giao diện trang này như sau

A screenshot of a computer

Description automatically generated

Hình 5.3.1.12: Hình ảnh khi mới mở trang Top-K Frequent Subgraph

Sau khi định nghĩa k và upload file phân tích thì sẽ ra được đồ thị được vẽ

A screenshot of a computer

Description automatically generated

Hình 5.3.1.13: Hình ảnh khi nhập đủ dữ liệu và thực hiện thuật toán cho trang Top-K Frequent Subgraph

### Vẽ đồ thị con thường xuyên với số lượng item

Dựa trên thuật toán TKG đã được nêu trên và một xíu chỉnh sửa khi bỏ đi các điều kiện loại bỏ các đồ thị con không phổ biến thì đã ra 1 trang mới như sau

A screenshot of a computer

Description automatically generated

Hình 5.3.2.14: Hình ảnh khi mới mở trang getSubgraphByNumberItems

Sau khi định nghĩa k và upload file phân tích thì sẽ ra được đồ thị được vẽ

A screenshot of a computer

Description automatically generated

Hình 5.3.2.15: Hình ảnh khi nhập đủ dữ liệu và thực hiện thuật toán cho trang getSubgraphByNumberItems

### Trang đề xuất sản phẩm thêm vào combo

Trang này sẽ sử dụng thuật toán TKG tìm tất cả Subgraph với những kỹ thuật cải tiến như Dynamic Search, Skip Stagegy và tiền xử lý thuật toán để gia tăng hiệu suất tính toán và sử dụng các thuật toán AER để tìm được quy luật liên quan của mẫu dữ liệu.

Việc đầu tiên cần làm là chọn file để phân tích

A screenshot of a computer

Description automatically generated

Hình 5.3.3.16: Hình ảnh khi mới mở trang findBestRecommendation

Sau khi file đã được chọn thì các lựa chọn để chọn dữ liệu đầu vào của combo hiển thị ra bên ngoài

A screenshot of a computer

Description automatically generated

Hình 5.3.3.17: Hình ảnh khi up file lên và nhấn submit để ra dữ liệu chọn

Người dùng nhấn chọn dữ liệu đầu vào của combo

A screenshot of a computer

Description automatically generated

Hình 5.3.3.18: Hình ảnh khi chọn dữ liệu

Sau khi chọn dữ liệu đầu vào cho combo và bấm nút find recommendation thì hệ thống bắt đầu tính toán và đưa ra cho kết quả có số liệu gần đúng nhất.

A screenshot of a computer

Description automatically generated

Hình 5.3.3.19: Hình ảnh khi nhấn Find Recommendation và có kết quả

Nếu như không đủ dữ liệu để thống kê kết quả thì hệ thống sẽ thông báo lỗi

A screenshot of a computer

Description automatically generated

Hình 5.3.3.20: Hình ảnh khi nhấn Find Recommendation và không có kết quả

# KẾT LUẬN

## Kết luận

### Số liệu đo đạc về hiệu suất

|  |  |
| --- | --- |
| Số dòng giao dịch | Thời gian(s) |
| 100 | 1.22 |
| 200 | 3.71 |
| 300 | 20.99 |
| 400 | 90 |
| 500 | 162 |

Bảng 6.1.1.2: Hình ảnh về bảng thời gian sau khi thử nghiệm trên các kích thước dữ liệu khác nhau

A graph with numbers and lines

Description automatically generated

Hình 6.1.1.21: Hình ảnh biểu đồ lý thuyết thể hiện mối quan hệ của dữ liệu số đồ thị xử lý và thời gain

Dựa vào số liệu đo hiệu suất của chương trình và số liệu lý thuyết đã được đo đạc vào năm 2020 ta có thể thấy được mặc dù đã được tối ưu nhưng thời gian để xử lý vẫn tăng đột biến khi ta bắt đầu tăng số lượng dữ liệu cần phải xử lý

### Dự đoán thời gian xử lý cho thuật toán

Dựa vào thời gian xử lý thuật toán trên ta có thể dự đoán thời gian thực hiện cho những số lượng sản phẩm tiếp theo từ đó có thể thấy hệ thống thực sự gặp khó khăn trong quá trình xử lý dữ liệu.

A graph with a line and a dotted line

Description automatically generated

Hình 6.1.2.22: Hình ảnh biểu đồ được dự đoán cho hệ thống

### Ưu điểm

* Đã hoàn thành được mục tiêu ban đầu khi đưa việc khai thác dữ liệu dạng đồ thị vào văn hóa bán hàng
* Hệ thống có thể đáp ứng cơ bản các cửa hàng vừa và nhỏ nơi có số lượng giao dịch mỗi tháng trên dưới 500 đơn hàng
* Là một đóng góp có sức ảnh hưởng cho việc nghiên cứu phát triển thuật toán khai thác dữ liệu dưới dạng đồ thị

### Khuyết điểm

* Vấn đề thời gian vẫn là chủ đề cốt lõi khi không thể đáp ứng về hiệu suất thời gian

🡺 Vấn đề mở rộng cực kỳ nan giải khi số lượng giao dịch để đáp ứng được cần phải cực lớn

* Vấn đề mở rộng lên hệ thống Website vẫn còn rất nan giải khi các hệ thống hầu hết chưa tối ưu hoàn toàn cho việc xử lý dữ liệu theo hướng đồ thị
* Hệ thống chỉ mới dừng lại ở việc thao tác trên file CSV chứ chưa thực sự thao tác trên database

## Hướng phát triển

* Tập trung phát triển về hiệu suất tính toán cho mô hình nhằm giải quyết vấn đề về mặt không gian
* Tiếp tục tìm hiểu các biện pháp tối ưu cho mô hình khi triển khai trên trình duyệt Website

# TÀI LIỆU THAM KHẢO

Tiếng Việt

…

Tiếng Anh

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-term Memory. *Neural Computation*, *9*, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention Is All You Need* (arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762