ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

HOMEWORK 02

Khai thác dữ liệu đồ thị

Sinh viên thực hiện: 21127229 - Dương Trường Bình

Giảng viên hướng dẫn: Lê Ngọc Thành
Lê Nhựt Nam

Lớp: 21KHDL

Mục lục

1 Problem 1

Câu hỏi: Thế nào là frequent patterns? Cho ví dụ? Tại sao chúng ta cần tìm frequent patterns trong dữ liệu?

Trả lời

- **Frequent patterns** (mẫu phổ biến) là các mẫu hoặc các tập hợp các mục (itemsets) xuất hiện thường xuyên trong dữ liệu (số lần xuất hiện lớn hơn hoặc bằng một ngưỡng support xác định trước). Frequent patterns thường được sử dụng để khai phá dữ liệu, giúp chúng ta hiểu rõ hơn về dữ liệu, từ đó có thể đưa ra các quyết định, dự đoán, hay khám phá kiến thức mới.
- **Ví dụ**
  - Trong một cửa hàng, một frequent pattern có thể là các mặt hàng thường được mua hoặc các tổ hợp mặt hàng thường được mua cùng nhau. Ví dụ: milk, bread, butter là một frequent pattern nếu nó xuất hiện nhiều lần trong dữ liệu.

    Giả sử chúng ta có một cơ sở dữ liệu (Bảng 1)về giao dịch mua sắm với mỗi giao dịch bao gồm các sản phẩm đã được mua và minSup được xác định là 3:

    Bảng 1: Danh sách các item

TID	Items
1	milk, bread, butter
2	milk, bread
3	milk, butter
4	milk, bread, butter
5	milk, bread

  - Trong trường hợp này, milk, bread là một frequent pattern vì nó xuất hiện 4 lần trong dữ liệu.
  - Trong một văn bản, một frequent pattern có thể là các từ hoặc cụm từ thường xuất hiện cùng nhau. Ví dụ: machine learning là một frequent pattern nếu nó xuất hiện nhiều lần trong văn bản.
- Chúng ta cần tìm frequent patterns trong dữ liệu vì chúng giúp chúng ta:
  1. Nâng cao hiểu biết về dữ liệu

- **Khám phá tri thức ẩn**: Frequent patterns hé lộ những mối liên hệ và xu hướng tiềm ẩn trong dữ liệu mà mắt thường khó có thể nhận ra. Ví dụ, việc phân tích dữ liệu bán hàng có thể cho thấy một số mặt hàng thường xuyên được mua cùng nhau, cung cấp thông tin chi tiết về sở thích mua sắm của khách hàng.
- **Phân đoạn dữ liệu hiệu quả**: Việc xác định các nhóm hoặc phân khúc khách hàng dựa trên frequent patterns hỗ trợ các chiến dịch tiếp thị và quảng cáo được nhắm mục tiêu hiệu quả hơn, tối ưu hóa hiệu quả sử dụng ngân sách.

2. Hỗ trợ quyết định chiến lược
   - **Dự đoán hành vi**: Frequent patterns có thể được sử dụng để xây dựng các mô hình dự đoán hành vi khách hàng, ví dụ như khả năng mua hàng tiếp theo, sản phẩm ưa thích, v.v. Giúp doanh nghiệp đưa ra các quyết định phù hợp về sản xuất, kinh doanh và cung cấp dịch vụ.
   - **Tối ưu hóa quy trình**: Phân tích frequent patterns trong dữ liệu quy trình sản xuất có thể giúp xác định các điểm nghẽn, lãng phí hoặc cơ hội cải tiến, từ đó tối ưu hóa hiệu quả hoạt động và giảm chi phí.
3. Cung cấp lợi thế cạnh tranh
   - **Tận dụng cơ hội thị trường**: Doanh nghiệp có thể khai thác frequent patterns để phát hiện các xu hướng thị trường mới, đưa ra sản phẩm, dịch vụ phù hợp, hoặc thâm nhập thị trường tiềm năng trước đối thủ cạnh tranh.
   - **Nâng cao trải nghiệm khách hàng**: Hiểu rõ hơn về hành vi và sở thích của khách hàng thông qua frequent patterns giúp doanh nghiệp cá nhân hóa trải nghiệm khách hàng, tăng mức độ hài lòng và lòng trung thành của họ.
4. Ngoài những lý do trên, việc tìm frequent patterns còn có các ứng dụng khác
   - **Phát hiện gian lận**: Phân tích patterns giao dịch bất thường có thể giúp phát hiện các hành vi gian lận trong tài chính, bảo hiểm, v.v.
   - **Chẩn đoán bệnh**: Xác định các patterns trong dữ liệu y tế có thể hỗ trợ chẩn đoán bệnh chính xác và sớm hơn.

2 Problem 2

Câu hỏi: Trình bày nguyên lý Apriori.

Trả lời

- **Nguyên tắc Apriori** là nền tảng cho thuật toán Apriori, giúp giảm thiểu độ phức tạp khi tìm kiếm các frequent itemsets và association rules. Nguyên tắc này bao gồm hai phần chính:
  - Tất cả các tập con của một frequent itemset cũng phải là frequent itemset.

    Ví dụ: Nếu $\{A, B, C\}$ là frequent itemset với minsup = 50%, thì tất cả các tập con của nó như $\{A, B\}, \{A, C\}, \{B, C\}$ cũng phải là frequent itemset với minsup = 50%.
  - Bất kỳ tập cha nào của một infrequent itemset cũng phải là infrequent itemset.

    Ví dụ: Nếu $\{X, Y, Z\}$ là infrequent itemset với minsup = 50%, thì tất cả các tập cha của nó như $\{X, Y, Z, W\}$ cũng phải là infrequent itemset với minsup = 50%.
- Thuật toán Apriori sử dụng **nguyên tắc Apriori** để lặp đi lặp lại tìm kiếm các frequent itemset với kích thước tăng dần. Quy trình hoạt động như sau:
  1. Khởi tạo
     - Xác định minsup (tỷ lệ hỗ trợ tối thiểu) và minconf (độ tin cậy tối thiểu).
     - Tìm tất cả các frequent itemset kích thước 1 (1-itemsets) bằng cách đếm số lần xuất hiện của từng item trong tập dữ liệu.
  2. Lặp:
     - Tạo các candidate k-itemsets (tập ứng viên) mới bằng cách kết hợp các frequent (k-1)-itemsets đã biết.
     - Prune: Loại bỏ các k-itemsets ứng viên mà có bất kỳ (k-1)-subset nào không phổ biến
     - Đếm số lần xuất hiện của các candidate itemsets trong cơ sở dữ liệu và chỉ giữ lại những k-itemsets có tần suất lớn hơn hoặc bằng ngưỡng tối thiểu.
     - Lặp lại quá trình cho đến khi không còn candidate itemsets nào hoặc không thể tạo ra candidate itemsets mới.
  3. Trích xuất frequent itemsets: Kết hợp tất cả các frequent itemsets đã tìm thấy trong các bước trước để tạo ra tất cả các frequent itemsets.
  4. Trích xuất association rules: Từ các frequent itemsets, tạo ra các association rules với một ngưỡng confidence cho trước.

3 Problem 3

Câu hỏi: Trình bày phương pháp của hai hướng tiếp cận Apriori và Pattern-growth. Mỗi hướng tiếp cận có những khó khăn gì?

Trả lời

3.1 Hướng tiếp cận Apriori

Phương pháp:

- Áp dụng nguyên tắc Apriori: một subgraph là frequent nếu tất cả các subgraph con của nó cũng frequent.
- Sử dụng chiến lược tìm kiếm theo chiều rộng: bắt đầu từ các subgraph nhỏ nhất (ví dụ: các đỉnh hoặc cạnh đơn lẻ) và lặp đi lặp lại để tạo ra các ứng cử viên lớn hơn bằng cách kết hợp các subgraph phổ biến từ lần lặp trước.

Thuật toán:

- **Khởi tạo:** Quét dữ liệu để tìm F_1, tập hợp tất cả các đồ thị con 1-cạnh (hoặc 1-đỉnh) phổ biến, cùng với độ trợ của chúng.
- **Lặp lại (k=2, 3, ... cho đến khi F_{k-1} rỗng):**
  - **Phát sinh ứng viên C_k:** Từ F_{k-1}, tạo ra tập hợp ứng viên các đồ thị con k-cạnh (hoặc k-đỉnh) bằng cách kết hợp các đồ thị con (k-1)-cạnh (hoặc (k-1)-đỉnh) phổ biến.
  - **Tỉa nhánh:** Loại bỏ các ứng viên không thỏa mãn tính chất bao đóng hướng xuống (downward closure)- mọi đồ thị con (k-1)-cạnh (hoặc (k-1)-đỉnh) của nó phải phổ biến.
  - **Tính độ phổ biến:** Quét dữ liệu để đếm số lần xuất hiện của mỗi ứng viên trong C_k.
  - $F_k = \{c \in C_k | count(c) \geq minSup\}$
- **Kết quả:** $F = F_1 \cup F_2 \cup ... \cup F_k$.

Hai nhánh của Apriori:

- **Vertex growing (AGM):** k là số đỉnh. Tập trung vào việc mở rộng đồ thị con bằng cách thêm đỉnh.

- **Edge growing (FGM):** k là số cạnh. Tập trung vào việc mở rộng đồ thị con bằng cách thêm cạnh.

Khó khăn:

- **Chi phí tính toán cao:** Việc kiểm tra đẳng cấu đồ thị (graph isomorphism) là một bài toán NP-Complete, dẫn đến chi phí tính toán cao cho cả ba bước: phát sinh ứng viên, tỉa nhánh và đếm độ phổ biến.
- **Phát sinh nhiều ứng viên:** Việc kết hợp hai đồ thị con k-cạnh có thể tạo ra nhiều ứng viên (k+1)-cạnh, làm tăng số lượng ứng viên cần kiểm tra.

3.2 Hướng tiếp cận Pattern-growth

Phương pháp: Mở rộng một đồ thị con phổ biến bằng cách thêm một cạnh mới và kiểm tra độ phổ biến của đồ thị mới. Phương pháp này tập trung vào việc mở rộng trực tiếp từ các đồ thị con phổ biến hiện có mà không cần phát sinh ứng viên.

Thuật toán:

- **Khởi tạo:** Bắt đầu từ một đồ thị con phổ biến (ví dụ: một đỉnh hoặc một cạnh).
- **Lặp lại:**
  - **Mở rộng:** Thêm một cạnh mới vào đồ thị con hiện tại.
  - **Kiểm tra độ phổ biến:** Kiểm tra xem đồ thị con mới có phổ biến hay không.
  - **Nếu phổ biến:** Tiếp tục mở rộng đồ thị con mới.
  - **Nếu không phổ biến:** Quay lại bước mở rộng và thử một cạnh khác.
- **Kết thúc:** Khi không thể mở rộng thêm được nữa.

Thuật toán gSpan: Một cải tiến của phương pháp Pattern-growth, giúp giảm thiểu việc tạo ra các đồ thị con trùng lặp và cải thiện việc kiểm tra đẳng cấu bằng cách sử dụng mã DFS (Depth-First Search).

- **Mã DFS:** Chuyển đổi đồ thị 2 chiều thành chuỗi các cạnh được mã hóa theo thứ tự duyệt DFS và chọn mã DFS nhỏ nhất để đại diện cho đồ thị.
- **Xây dựng không gian tìm kiếm:** Tổ chức các mã DFS theo cấu trúc cây để dễ dàng tìm kiếm và so sánh.

Khó khăn:

- **Đồ thị trùng lặp:** Có thể tạo ra nhiều lần cùng một đồ thị con do thứ tự duyệt và lựa chọn điểm bắt đầu khác nhau.
- **Kiểm tra đẳng cấu:** Vẫn cần thực hiện kiểm tra đẳng cấu để đảm bảo tính duy nhất của các đồ thị con được tìm thấy.

3.3 Tóm lại

Apriori:

- **Ưu điểm:** Dễ triển khai và dễ hiểu. Phù hợp với các tập dữ liệu nhỏ và không phức tạp.
- **Nhược điểm:** Tốn kém tính toán và không hiệu quả với dữ liệu lớn do số lượng mẫu kết hợp tăng theo cấp số nhân. Cần kiểm tra đẳng cấu nhiều lần, gây tốn thời gian và tài nguyên.

Pattern-growth:

- **Ưu điểm:** Hiệu quả hơn với dữ liệu lớn, giảm thiểu số lượng mẫu kết hợp cần kiểm tra. Tốc độ nhanh hơn do không cần phát sinh ứng viên. Có thể tận dụng cấu trúc đồ thị hiện có để mở rộng trực tiếp.
- **Nhược điểm:** Phức tạp trong việc triển khai và quản lý bộ nhớ. Cần tối ưu hóa để xử lý các đồ thị lớn. Cũng gặp khó khăn trong việc kiểm tra đẳng cấu và có thể tạo ra các đồ thị con trùng lặp.

4 Problem 4

Câu hỏi Với MSSV 21127229: Trình bày lại gSpan và cho ví dụ minh họa tính toán. Và trình bày về tiếp cận tham lam Subdue. Phân biệt, đánh giá ưu điểm và nhược điểm của từng loại tiếp cận

Trả lời

4.1 gSpan (graph-based Substructure pattern mining)

Mô tả gSpan: gSpan là một thuật toán khai thác mẫu đồ thị con thường xuyên hiệu quả trong tập dữ liệu đồ thị lớn, thuộc nhóm phương pháp tìm mẫu phổ biến dựa trên chiều sâu (Depth-based).

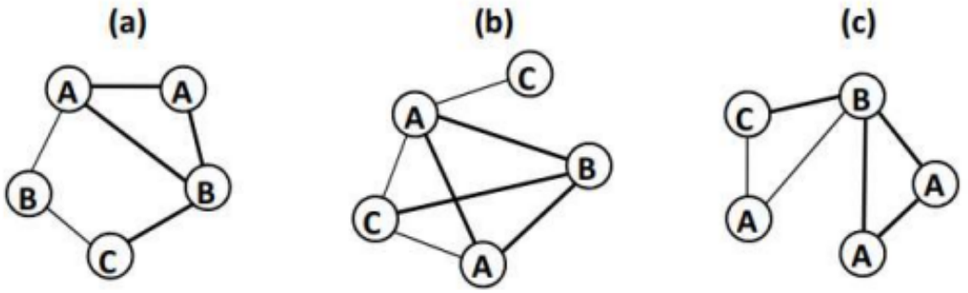
- **Mục tiêu:**

  - gSpan nhắm vào việc giảm thiểu hoặc loại bỏ chi phí tạo ứng cử viên và loại bỏ các dương tính giả.
  - Khám phá tất cả các đồ thị con thường xuyên mà không cần tạo ứng cử viên và không cần loại bỏ dương tính giả.

- **Cơ chế hoạt động:**
  - **Mã hóa DFS:** gSpan xây dựng một thứ tự từ điển mới giữa các đồ thị bằng cách ánh xạ mỗi đồ thị tới một mã DFS (Depth-First Search) tối thiểu duy nhất đóng vai trò là nhãn chính tắc của nó.
  - **Tìm kiếm theo chiều sâu:** Dựa trên thứ tự từ điển này, gSpan áp dụng chiến lược tìm kiếm theo chiều sâu (DFS) để khai thác các đồ thị con được kết nối thường xuyên một cách hiệu quả.
  - **Cây Mã DFS:** gSpan sử dụng Cây Mã DFS để tổ chức các mã DFS theo thứ tự từ điển, cho phép tìm kiếm và loại bỏ trùng lặp hiệu quả.
- **Ưu điểm:**
  - Hiệu suất cao hơn các thuật toán trước đó, đôi khi hơn hẳn một bậc.
  - Khả năng khai thác các đồ thị con lớn và thường xuyên trong tập dữ liệu đồ thị lớn với mức hỗ trợ tối thiểu thấp hơn so với các nghiên cứu trước đây.
- **Nhược điểm:**
  - Hiệu suất của gSpan có thể bị ảnh hưởng bởi độ sâu của cây tìm kiếm, đặc biệt là khi xử lý các đồ thị có cấu trúc phức tạp và kích thước rất lớn.

4.2 Ví dụ Minh Họa Tính Toán với gSpan

Giả sử chúng ta có ba đồ thị đầu vào như hình minh họa và mức hỗ trợ tối thiểu là 3.



Hình 4.2.1: Ví dụ minh họa tính toán của gSpan với mức hỗ trợ tối thiểu là 3

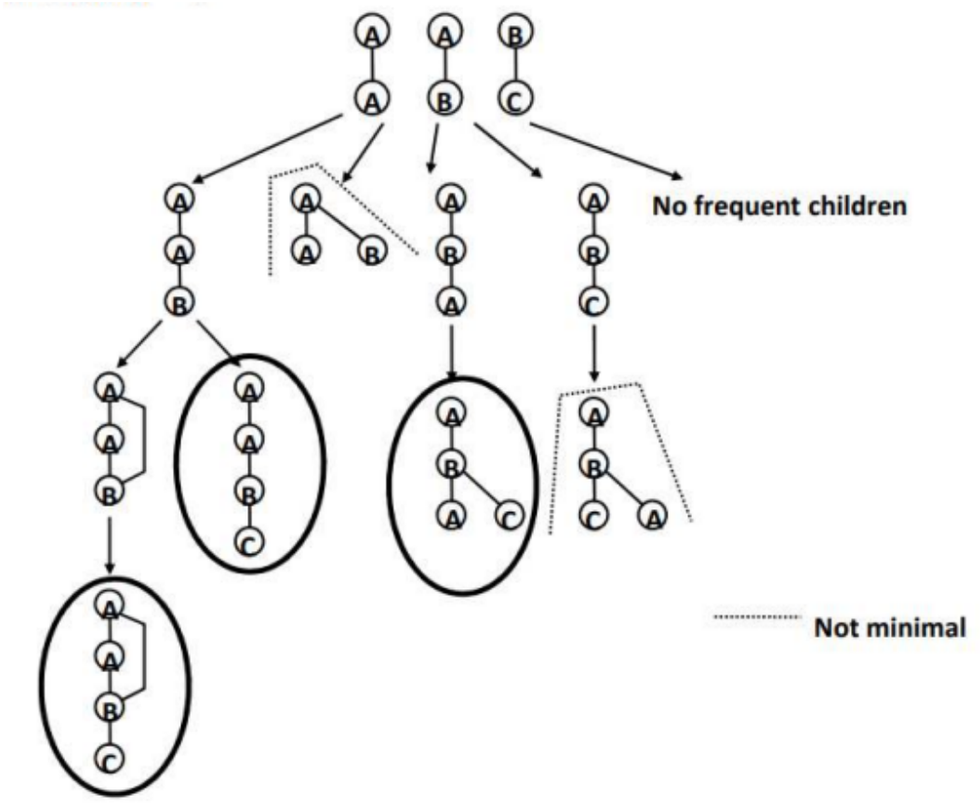
Bước 1: Xây dựng mã DFS cho từng đồ thị

Đầu tiên, chúng ta mã hóa từng đồ thị bằng cách sử dụng mã DFS. Giả sử mã DFS của các đồ thị là:

- **Đồ thị (a):** (A, B), (A, C), (B, C), (C, D)
- **Đồ thị (b):** (A, B), (A, C), (A, D), (B, D), (C, D)
- **Đồ thị (c):** (A, B), (A, C), (A, D), (B, D)

Bước 2: Xây dựng Cây Mã DFS

Dựa trên mã DFS, chúng ta xây dựng Cây Mã DFS để tổ chức và tìm kiếm các đồ thị con phổ biến.



Bước 3: Khai thác các đồ thị con thường xuyên

Dựa vào Cây Mã DFS, chúng ta tiến hành tìm kiếm các đồ thị con thường xuyên với mức hỗ trợ tối thiểu là 3. Cụ thể:

- **Mã DFS: (A-B)**
  - Xuất hiện trong: Đồ thị (a), Đồ thị (b), Đồ thị (c)
  - Mức hỗ trợ: 3 (thỏa mãn mức hỗ trợ tối thiểu)
- **Mã DFS: (A-C)**
  - Xuất hiện trong: Đồ thị (a), Đồ thị (b), Đồ thị (c)
  - Mức hỗ trợ: 3 (thỏa mãn mức hỗ trợ tối thiểu)
- **Mã DFS: (A-B, A-C)**
  - Xuất hiện trong: Đồ thị (a), Đồ thị (b), Đồ thị (c)
  - Mức hỗ trợ: 3 (thỏa mãn mức hỗ trợ tối thiểu)
- **Mã DFS: (A-B, B-D)**

  - Xuất hiện trong: Đồ thị (b), Đồ thị (c)
  - Mức hỗ trợ: 2 (không thỏa mãn mức hỗ trợ tối thiểu)

- **Mã DFS: (A-C, C-D)**
  - Xuất hiện trong: Đồ thị (a), Đồ thị (b)
  - Mức hỗ trợ: 2 (không thỏa mãn mức hỗ trợ tối thiểu)

Bước 4: Loại bỏ các mã không thường xuyên và không tối thiểu

Dựa trên mức hỗ trợ tối thiểu, chúng ta loại bỏ các đồ thị con không thường xuyên và không tối thiểu. Ví dụ, mã (A-B, B-D) và (A-C, C-D) không thỏa mãn mức hỗ trợ tối thiểu, do đó sẽ bị loại bỏ.

Bước 5: Lưu trữ và đánh giá các đồ thị con phổ biến

Các đồ thị con phổ biến sau khi loại bỏ sẽ được lưu trữ và đánh giá:

- (A-B)
- (A-C)
- (A-B, A-C)

Các đồ thị con này đều thỏa mãn mức hỗ trợ tối thiểu và là các mẫu đồ thị con phổ biến trong tập dữ liệu.

4.3 Tiếp Cận Tham Lam Subdue

Mô tả Subdue: Subdue là một thuật toán tham lam để tìm một số đồ thị con phổ biến nhất, thuộc nhóm phương pháp tìm mẫu phổ biến tham lam.

- **Mục tiêu:**
  - Tìm kiếm các mẫu đồ thị con phổ biến bằng cách nén đồ thị sử dụng các mẫu đồ thị.
- **Cơ chế hoạt động:**
  - **Nén đồ thị:** Subdue hoạt động dựa trên độ dài mô tả bằng cách nén các đồ thị bằng cách sử dụng các mẫu đồ thị. Mẫu được sử dụng để nén tối đa được coi là mẫu phổ biến.
  - **Tìm kiếm theo chùm tia:** Subdue sử dụng kỹ thuật tìm kiếm theo chùm tia (Beam Search), tương tự như BFS (Breadth-First Search), nhưng chỉ duyệt qua nút tốt nhất

ở mỗi cấp, bỏ qua các nút khác.

- **Ưu điểm:**
  - **Nhanh chóng:** Quá trình thực thi nhanh chóng do tính chất tham lam của thuật toán.
- **Nhược điểm:**
  - **Không đầy đủ:** Subdue có thể không tìm thấy tất cả các đồ thị con phổ biến.
  - **Độ chính xác phụ thuộc vào tham số:** Hiệu quả của Subdue phụ thuộc vào việc lựa chọn tham số, ví dụ như chiều rộng chùm tia trong tìm kiếm theo chùm tia.

4.4 Phân Biệt, Đánh Giá Ưu Điểm và Nhược Điểm

1. gSpan (graph-based Substructure pattern mining):

- **Loại thuật toán:** Khai thác mẫu đồ thị con thường xuyên dựa trên chiều sâu (Depth-based).
- **Cơ chế hoạt động:**
  - **Không tạo ứng cử viên:** gSpan khám phá các đồ thị con thường xuyên mà không cần tạo ra các ứng cử viên.
  - **Mã hóa DFS:** gSpan xây dựng một mã DFS tối thiểu duy nhất cho mỗi đồ thị, đóng vai trò là nhãn chính tắc của nó.
  - **Tìm kiếm theo chiều sâu:** gSpan sử dụng chiến lược tìm kiếm theo chiều sâu (DFS) dựa trên thứ tự từ điển của các mã DFS để khai thác các đồ thị con được kết nối thường xuyên.
  - **Cây Mã DFS:** gSpan tổ chức các mã DFS trong một cây, cho phép tìm kiếm và loại bỏ trùng lặp hiệu quả.
- **Ưu điểm:**
  - **Hiệu suất cao:** gSpan thường có hiệu suất cao hơn các thuật toán trước đó, đôi khi hơn hẳn một bậc.
  - **Đầy đủ:** gSpan có khả năng khai thác đầy đủ các đồ thị con lớn và thường xuyên trong tập dữ liệu đồ thị lớn.
- **Nhược điểm:**

- **Hiệu suất phụ thuộc vào cấu trúc đồ thị:** Do dựa trên tìm kiếm theo chiều sâu, hiệu suất của gSpan có thể bị ảnh hưởng bởi độ sâu của cây tìm kiếm, đặc biệt là khi xử lý các đồ thị có cấu trúc phức tạp và kích thước rất lớn.

2. Subdue (Greedy Approach):

- **Loại thuật toán:** Khai thác mẫu đồ thị con phổ biến dựa trên phương pháp tham lam.
- **Cơ chế hoạt động:**
  - **Nén đồ thị:** Subdue hoạt động dựa trên độ dài mô tả bằng cách nén các đồ thị bằng cách sử dụng các mẫu đồ thị.
  - **Tìm kiếm theo chùm tia:** Subdue sử dụng kỹ thuật tìm kiếm theo chùm tia (Beam Search), tương tự như BFS (Breadth-First Search), nhưng chỉ duyệt qua nút tốt nhất ở mỗi cấp, bỏ qua các nút khác.
- **Ưu điểm:**
  - **Nhanh chóng:** Quá trình thực thi nhanh chóng do tính chất tham lam của thuật toán.
- **Nhược điểm:**
  - **Không đầy đủ:** Subdue có thể không tìm thấy tất cả các đồ thị con phổ biến.
  - **Độ chính xác phụ thuộc vào tham số:** Hiệu quả của Subdue phụ thuộc vào việc lựa chọn tham số, ví dụ như chiều rộng chùm tia trong tìm kiếm theo chùm tia.

3. Phân Biệt Chi Tiết giữa gSpan và Subdue:

Tiêu chí	**gSpan**	**Subdue**
Loại thuật toán	Pattern-growth (dựa trên chiều sâu)	Tham lam
Tìm kiếm	Chiều sâu (DFS)	Chùm tia (Beam Search)
Mục tiêu	Khai thác đầy đủ các đồ thị con thường xuyên	Tìm các đồ thị con phổ biến nhất
Ưu điểm	- Hiệu suất cao hơn các thuật toán trước đó - Đầy đủ	- Thực thi nhanh chóng - Dễ triển khai
Nhược điểm	- Hiệu suất phụ thuộc vào cấu trúc đồ thị - Khó khăn với đồ thị lớn	- Không đầy đủ - Phụ thuộc vào tham số

Bảng 2: So sánh giữa gSpan và Subdue

Tài liệu tham khảo

[1] Thuật Toán Apriori - Khai Thác Luật Kết Hợp Trong Data Mining, Viblo, https://viblo.asia/p/thuat-toan-apriori-khai-pha-luat-ket-hop-trong-data-mining-3P0lPEv85ox, truy cập ngày 20/07/2024.

[2] Slide của thầy Lê Ngọc Thành, *Topic 05 - Frequent Graph Pattern Mining*, Trường Đại học Khoa học Tự nhiên, 2024