BỘ THÔNG TIN VÀ TRUYỀN THÔNG
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**HOÀNG MINH QUANG**

# NGHIÊN CỨU, PHÁT TRIỂN MỘT SỐ PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU TRÊN DỮ LIỆU CÓ CẤU TRÚC

LUẬN ÁN TIẾN SĨ KỸ THUẬT

Hà Nội – Năm 2020

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**HOÀNG MINH QUANG**

# NGHIÊN CỨU, PHÁT TRIỂN MỘT SỐ PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU TRÊN DỮ LIỆU CÓ CẤU TRÚC

Chuyên ngành : Hệ thống thông tin
Mã số: 09.48.01.04

LUẬN ÁN TIẾN SĨ KỸ THUẬT

NGƯỜI HƯỚNG DẪN KHOA HỌC:
1. GS. TS. VŨ ĐỨC THI
2. GS. TSKH. NGUYỄN NGỌC SAN

Hà Nội - Năm 2020

# LỜI CẢM ƠN

Đầu tiên, nghiên cứu sinh xin được gửi lời cảm ơn sâu sắc tới hai người thầy hướng dẫn; GS. TS. Vũ Đức Thi và GS. TSKH. Nguyễn Ngọc San đã định hướng nghiên cứu và chỉ dẫn các giải pháp khoa học trong cả quá trình nghiên cứu sinh thực hiện luận án.

Nghiên cứu sinh xin gửi lời cảm ơn tới lãnh đạo và tập thể cán bộ Viện Công nghệ thông tin, Viện Hàn lâm Khoa học và Công nghệ Việt nam cùng phòng Khoa học dữ liệu và Ứng dụng nơi nghiên cứu sinh đang công tác. Nghiên cứu sinh cũng chân thành gửi lời cảm ơn tới TS. Nguyễn Việt Anh đã đọc và góp ý vào phiên bản dự thảo của luận án.

Nghiên cứu sinh xin cảm ơn lãnh đạo, các nhà khoa học Học viện Công nghệ Bưu chính viễn thông đã tạo điều kiện, trợ giúp nghiên cứu sinh trong quá trình thực hiện luận án. Nghiên cứu sinh cũng xin cảm ơn các bạn bè, đồng nghiệp, các nhà khoa học đã có những đóng góp quý báu cho luận án.

Nghiên cứu sinh xin cảm ơn Cha, Mẹ đã động viên khuyến khích nghiên cứu sinh trong quá trình nghiên cứu học tập. Cảm ơn vợ Bùi Thị Thuý Hà và hai con Hoàng Hải Lâm và Hoàng Minh Thư, những hy sinh trong quá trình nghiên cứu sinh thực hiện luận án đã tạo động lực để nghiên cứu sinh cố gắng phấn đấu đến ngày hôm nay.

# LỜI CAM ĐOAN

Nghiên cứu sinh xin cam đoan những công trình công bố trong luận án này là kết quả của nghiên cứu sinh nghiên cứu dưới sự hướng dẫn khoa học của GS. TS. Vũ Đức Thi và GS. TSKH. Nguyễn Ngọc San. Những kết quả được nghiên cứu sinh trình bày trong luận án này là mới, duy nhất và chưa từng được công bố trong bất kỳ công trình nào khác.

Nghiên cứu sinh xin hoàn toàn chịu trách nhiệm trước lời cam đoan của mình.

*Hà Nội, ngày 31 tháng 12 năm 2019.*

Nghiên cứu sinh

**Hoàng Minh Quang**

# MỤC LỤC

# DANH MỤC HÌNH VẼ

# DANH MỤC BẢNG BIỂU

# DANH MỤC THUẬT NGỮ

| Thuật ngữ tiếng Anh | Thuật ngữ tiếng Việt |
|---|---|
| antikey | phản khóa |
| antisymmetry | phản đối xứng |
| attribute | thuộc tính |
| attribute reduct | rút gọn thuộc tính |
| belief function | hàm niềm tin |
| $\beta$ lower distribution reduct | rút gọn phân phối cận dưới $\beta$ |
| $\beta$ upper distribution reduct | rút gọn phân phối cận trên $\beta$ |
| binary relation | quan hệ hai ngôi |
| boudary | vùng biên |
| capacity | sức chứa |
| closed frequent subgraph | đồ thị con thường xuyên đóng |
| closed set | tập đóng |
| closure | đóng |
| closure system | hệ đóng |
| commonality function | hàm tính chất chung |
| complete lattice | dàn giao khái niệm |
| concept lattice | dàn giao khái niệm |
| conjugate | liên hợp |
| consistent | nhất quán |
| co-Möbius transform | đồng biến đổi Möbius |
| data mining | khai phá dữ liệu |
| decision table | bảng quyết định |
| Dempster's rule of combination | luật kết hợp Dempster |
| domain value | miền giá trị |
| discernibility matrix | ma trận phân biệt |

| | |
|---|---|
| equality set | tập bằng nhau |
| equivalent class | lớp tương đương |
| extent | phạm vi |
| plausibility function | hàm sự thật |
| frame of discernment | khung phân biệt |
| frequent subgraph | đồ thị con thường xuyên |
| focal element | phần tử tiêu điểm |
| formal concept | khái niệm chính thức |
| formal concept analysis (FCA) | phân tích khái niệm chính thức |
| formal context | ngữ cảnh chính thức |
| full family | họ đầy đủ |
| f-family | họ f |
| functional dependency | phụ thuộc hàm |
| Galois connection | kết nối Galois |
| graph | đồ thị |
| graph datatabase | cơ sở dữ liệu đồ thị |
| graph edit distance | khoảng cách sửa đổi đồ thị |
| greatest lower bound | lớn nhất cận dưới |
| indiscernibility relation | quan hệ bất khả phân biệt |
| information function | hàm thông tin |
| information system | hệ thông tin |
| intent | ý định |
| interval | khoảng |
| isomorphism | đẳng cấu |
| isomorphism subgraph | đẳng cấu đồ thị con |
| key | khóa |

| | |
|---|---|
| k-subgraph | k-đồ thị con |
| labeled graph | đồ thị được gắn nhãn |
| lattice | dàn giao |
| least upper bound | nhỏ nhất cận trên |
| lexicographic order | thứ tự quy định |
| locally finite | hữu hạn cục bộ |
| lower approximation | xấp xỉ dưới |
| mass allocation function | hàm cấp phát khối |
| maximal commonality subgraph | đồ thị con chung lớn nhất |
| maxmimal equality system | hệ bằng nhau cực đại |
| minimal key | khóa tối tiểu |
| monotonicity | đơn điệu |
| Möbius transform | biến đổi Möbius |
| multi-label classification | phân loại đa nhãn |
| object | đối tượng |
| ordered set | tập có thứ tự |
| partial order | thứ tự bộ phận |
| partially ordered set | tập thứ tự bộ phận |
| partition | phân hoạch |
| positive region | miền dương |
| powerset | tập tất cả các tập |
| reduct | rút gọn |
| relfexivity | phản xạ |
| relation | quan hệ |
| relational database | cơ sở dữ liệu quan hệ |
| relational scheme | lược đồ quan hệ |

| | |
|---|---|
| rough set | tập thô |
| set system | hệ thống tập hợp |
| shortest path kernels | các nhân đường đi ngắn nhất |
| Sperner system | hệ Sperner |
| subconcept-superconcept relation | quan hệ khái niệm con - khái niệm cha |
| subgraph | đồ thị con |
| subset | tập con |
| supergraph | đồ thị cha |
| theory | lý thuyết |
| transitivity | bắc cầu |
| universal set | tập vũ trụ |
| upper approximation | xấp xỉ trên |
| variable precision rough set | tập thô chính xác biến |

# LỜI MỞ ĐẦU

## 1. TỔNG QUAN LUẬN ÁN VÀ LÝ DO CHỌN ĐỀ TÀI

Khai phá dữ liệu lớn [57] là một xu hướng phát triển công nghệ mang tính cách mạng, ngày càng được ứng dụng rộng rãi, và đặc biệt còn nhiều tiềm năng phát triển trên toàn thế giới. Trong báo cáo [13], dữ liệu lớn được được định nghĩa là "các công nghệ dữ liệu lớn mô tả một thế hệ công nghệ và kiến trúc mới được thiết kế để trích xuất các giá trị từ các khối lượng dữ liệu rất lớn và đa dạng bằng cách phân tích, khám phá ở tốc độ cao"[101]. Khai phá dữ liệu lớn có thể được ứng dụng để cải tiến công nghệ ở nhiều lĩnh vực quan trọng như: y tế, giao thông, tài chính, giáo dục, [28], [63] nhằm đem lại lợi ích trong việc hỗ trợ ra quyết định, cắt giảm chi phí, và tạo ra các sản phẩm, dịch vụ mới.

Mặc dù việc khai phá dữ liệu lớn đem lại giá trị to lớn và ý nghĩa, tuy nhiên, đây cũng là một lĩnh vực đòi hỏi công nghệ cao, đầu tư lớn, với nhiều thách thức. Nguyên nhân xuất phát từ hai đặc trưng cơ bản của dữ liệu lớn, đó là: tính lớn và tính đa dạng, phức tạp. Do độ lớn của dữ liệu, việc khai phá thường mất nhiều thời gian và chi phí, độ phức tạp tính toán của khai phá dữ liệu lớn thường là độ phức tạp hàm mũ. Hơn nữa, vì dữ liệu lớn và phức tạp, nên việc khai phá dữ liệu cần trích xuất được các thông tin cốt lõi để khai phá, thay vì xử lý cả tập hợp dữ liệu lớn, có nhiều dữ liệu dư thừa, không mang giá trị hữu ích. Do vậy, vấn đề cơ bản của xử lý dữ liệu lớn là cải tiến tốc độ xử lý dữ liệu và tăng giá trị của dữ liệu được khai phá.

Dữ liệu lớn, trước hết là độ lớn của dữ liệu được thu thập, tập hợp và lưu trữ trên một cụm hệ thống máy tính phân tán, không thể lưu trữ trên các máy tính độc lập, khó có thể khai phá được các giá trị tiềm ẩn. Với dữ liệu lớn, thời gian truy xuất thông tin trong cả hai việc đọc và viết đều gặp khó khăn với một độ trễ cao. Nếu không tối ưu thời gian truy xuất, giảm tập hợp dữ liệu thành một tập con thì khó xử lý do khai phá

dữ liệu yêu cầu phải đáp ứng trong thời gian nhất định không thể kéo dài hơn. Chẳng hạn không thể khai phá dữ liệu phòng chống xâm nhập máy tính trái phép trong khi việc truy xuất dữ liệu đã mất hàng tiếng đồng hồ chưa kể thời gian khai phá dữ liệu. Lúc đó kết quả khai phá dữ liệu sẽ không có ý nghĩa vì tội phạm đã vượt qua hệ thống an ninh và kịp gây ra tất cả tác động xấu.

Liên quan đến tính đa dạng của dữ liệu. Dữ liệu được thu thập từ nhiều nguồn nên kiểu biểu diễn của dữ liệu khác nhau. Dữ liệu lớn được thu thập từ nhiều nguồn dữ liệu khác nhau như các hệ thống quản trị dữ liệu, mạng internet, mạng xã hội, kênh thông tin giải trí, truyền hình kỹ thuật số, các thiết bị truyền thông đa phương tiện, các thiết bị di dộng, các thiết bị vạn vật kết nối v.v. đã tạo ra tập hợp dữ liệu đa dạng kiểu mà các thuật toán khai phá dữ liệu chưa thể áp dụng được. Mỗi thuật toán khai phá dữ liệu chỉ có thể khai phá dữ liệu trên một tập hợp dữ liệu thống nhất về kiểu dạng biểu diễn. Do vậy, trước khi khai phá dữ liệu thì tập hợp dữ liệu phải được đưa về chung kiểu biểu diễn. Sau đó các kiểu biểu diễn phải được biến đổi về một dạng cấu trúc dữ liệu chung đồng nhất. Theo một số công trình nghiên cứu, dữ liệu có thể được phân chia vào ba kiểu dữ liệu là dữ liệu có cấu trúc, dữ liệu bán cấu trúc và dữ liệu phi cấu trúc dựa trên biểu diễn từ việc thu thập từ các nguồn dữ liệu. Dữ liệu có cấu trúc được hình dung như một lược đồ có sẵn cho một tập hợp dữ liệu. Dữ liệu bán cấu trúc có một phần lược đồ định trước và một phần không có lược đồ định trước. Dữ liệu phi cấu trúc là dữ liệu không có lược đồ định kiểu dữ liệu trước. Có thể lấy ví dụ dữ liệu có cấu trúc là dữ liệu dạng bảng trong các hệ quản trị cơ sở dữ liệu quan hệ, dữ liệu phi cấu trúc là dữ liệu không có bất kỳ lược đồ định nghĩa nào trước như âm thanh, hình ảnh, video, văn bản tự do, email và dữ liệu bán cấu trúc là dữ liệu xml có các đỉnh là lược đồ định trước còn thông tin mô tả là các dữ liệu không có lược đồ định trước.

Các công trình khoa học đã chỉ ra rằng dù dữ liệu có lược đồ định trước như các biểu diễn cấu trúc dữ liệu dạng bảng hay các dữ liệu không có lược đồ định trước như

âm thanh, hình ảnh, video, văn bản,... thì tuỳ vào đặc trưng của dữ liệu và mục tiêu cần khai phá, các tập hợp dữ liệu đều có khả năng sử dụng một kiểu biểu diễn dạng đồ thị [35], [94], [95] để giải quyết vấn đề trong khai phá dữ liệu. Biểu diễn dữ liệu đồ thị là biểu diễn phức tạp nhất về dữ liệu, có thể được coi là biểu diễn dữ liệu có cấu trúc thông qua lược đồ định kiểu của đỉnh và cạnh trong đồ thị. Tương quan với biểu diễn đồ thị phức tạp là thời gian khai khá rất chậm, chẳng hạn như trên các dữ liệu biểu diễn dạng đồ thị như biểu diễn các cấu trúc hoá học, các cấu trúc sinh học, các mạng máy tính và mạng xã hội. Các thuật toán khai phá trên dữ liệu đồ thị hầu hết đều nằm trong vùng độ phức tạp thời gian không đa thức thậm chí có thể lên đến độ phức tạp thời gian hàm mũ. Để khai phá được các tập dữ liệu có biểu diễn cấu trúc đồ thị thì cần phải đưa ra các điều kiện ràng buộc để đưa về độ phức tạp thời gian đa thức.

Với ý nghĩa thực tiễn to lớn của ngành khai phá dữ liệu lớn, nhiều công trình khoa học đã tập trung nghiên cứu, phát triển các thuật toán nhằm cải tiến việc xử lý dữ liệu. Một số hướng nghiên cứu chính của các nhà khoa học trên thế giới trong việc khai phá dữ liệu như sau: đánh chỉ mục và truy vấn dữ liệu [90], [91], tìm kiếm theo từ khóa [18], [54], so khớp đồ thị [15], [16], mô tả đồ thị lớn [2], [38], [77], khai phá các mẫu thường xuyên [3], [7], [14], [26], [37], [39], [41], [44], [45], [46], [47], [49], [52], [55], [81], [89], [90], phân cụm dữ liệu [1], [5], [8], [24], [27], phân lớp dữ liệu [10], [23], [24], [33], [34], [43], [50], [70], [71], [82], [83], [85], [97], [98], [99], khai phá các dữ liệu phát triển theo thời gian [4], [6], [7], [9], [53].

Trong luận án này, nghiên cứu sinh tập trung vào cả hai bài toán cơ bản của ngành xử lý dữ liệu lớn là: tăng giá trị của dữ liệu và tăng tốc độ xử lý dữ liệu. Kết quả của luận án giúp nâng cao tính hiệu quả và giảm chi phí của việc khai phá dữ liệu lớn. Cụ thể, nghiên cứu sinh tập trung nghiên cứu, giải quyết hai bài toán: (i) một là các bài toán liên quan đến rút gọn thuộc tính, rút gọn đối tượng, giảm dữ liệu dư thừa, trích xuất được những dữ liệu nhỏ, đặc trưng, chính xác hơn, nhằm xác định giá trị cốt lõi

trong tập hợp dữ liệu lớn và phức tạp, (ii) hai là bài toán tối ưu hóa tính toán, cải thiện tốc độ và chi phí tính toán trong khai phá dữ liệu có độ phức tạp tính toán lớn như độ phức tạp tính toán hàm mũ hay độ phức tạp tính toán trong thời gian không đa thức.

## 2. MỤC TIÊU - ĐỐI TƯỢNG - PHẠM VI NGHIÊN CỨU

### Mục tiêu nghiên cứu

Đặt mục tiêu giải quyết hai bài toán trên, nghiên cứu sinh nghiên cứu, phát triển một số phương pháp khai phá dữ liệu trên dữ liệu có cấu trúc, tập trung vào dữ liệu biểu diễn cấu trúc dạng bảng và dạng đồ thị. Đối với dữ liệu dạng bảng, mục tiêu nghiên cứu là các bài toán giảm dư thừa dữ liệu, rút gọn thuộc tính, rút gọn đối tượng để thu được tập dữ liệu nhỏ hơn trong khi vẫn bảo toàn được tính chất rút gọn thuộc tính, sinh cây quyết định trong khai phá dữ liệu lớn. Đối với biểu diễn dữ liệu dạng đồ thị, mục tiêu nghiên cứu là tối ưu tính toán các bài toán có độ phức tạp thời gian không đa thức xuống thời gian đa thức sử dụng một số ràng buộc dữ liệu để có thể khám phá tri thức từ dữ liệu trong thời gian chấp nhận được và các bài toán liên quan đến khai phá các tập dữ liệu mà dạng biểu diễn đồ thị còn gặp khó khăn trong khi đối với các dạng biểu diễn dữ liệu khác đã có phương pháp thực hiện.

### Đối tượng nghiên cứu

Trong luận án này, nghiên cứu sinh đặt trọng tâm khai phá dữ liệu trên biểu diễn dữ liệu có cấu trúc dạng bảng quyết định nhất quán và biểu diễn đồ thị của cơ sở dữ liệu đồ thị như biểu diễn dữ liệu cấu trúc hóa học, biểu diễn dữ liệu sinh học, biểu diễn dữ liệu mạng máy tính, mạng xã hội. Trên tập dữ liệu được lựa chọn, nghiên cứu sinh phát triển một số thuật toán phục vụ khai phá dữ liệu lớn như giảm dư thừa, rút gọn dữ liệu hoặc tối ưu tính toán về độ phức tạp thời gian đa thức để đáp ứng thời gian khai phá dữ liệu cho phép đối với các thuật toán mà thông thường cần giải quyết trong độ phức tạp thời gian không đa thức.

**Phạm vi nghiên cứu**

Luận án tập trung vào hai đối tượng với phạm vi như: (i) bảng quyết định nhất quán với các bài toán tìm một rút gọn thuộc tính không heuristic, tìm một rút gọn đối tượng và sinh cây quyết định, và (ii) cơ sở dữ liệu giao tác đồ thị với bài toán khai phá đồ thị con thường xuyên đóng và phân loại đồ thị đa nhãn.

## 3. KẾT QUẢ - Ý NGHĨA KHOA HỌC VÀ THỰC TIỄN

Trong luận án, nghiên cứu sinh đã nghiên cứu cải tiến một số phương pháp khai phá dữ liệu đối với biểu diễn dữ liệu có cấu trúc dạng bảng và dạng đồ thị. Các kết quả đạt được bao gồm:

1. *Nghiên cứu rút gọn thuộc tính bảng quyết định nhất quán* Tìm được một rút gọn thuộc tính trong thời gian đa thức không sử dụng heuristic như các phương pháp tìm một rút gọn thuộc tính khác.

2. *Nghiên cứu rút gọn đối tượng bảng quyết định nhất quán* Tìm được một rút gọn đối tượng trong thời gian đa thức mà vẫn bảo toàn quá trình tìm tất cả các rút gọn thuộc tính.

3. *Nghiên cứu cây quyết định* Cải tiến phương pháp sinh cây quyết định thực hiện nhanh hơn quá trình sinh cây quyết định của thuật toán ID3.

4. *Nghiên cứu khai phá đồ thị con thường xuyên đóng* Chứng minh vấn đề đẳng cấu đồ thị con giải quyết trong thời gian đa thức trong khai phá đồ thị con thường xuyên đóng trong khi các thuật toán khai phá đồ thị con thường xuyên đóng khác chưa giải quyết được vấn đề đẳng cấu đồ thị con trong thời gian đa thức.

5. *Nghiên cứu phân loại đa nhãn cho đồ thị* Xây dựng độ đo trên dàn giao khái

niệm áp dụng cho phân loại đa nhãn đồ thị sử dụng lý thuyết Dempster-Shafer, trong khi các công trình phân loại đa nhãn theo lý thuyết Dempster-Shafer khác phải xây dựng độ đo dựa trên biểu diễn véctơ mà đồ thị không có biểu diễn véctơ.

Các kết quả nghiên cứu của nghiên cứu sinh đều có chứng minh tính đúng đắn và đầy đủ đã thể hiện ý nghĩa khoa học của luận án. Ngoài ra, các kết quả này có thể áp dụng cho cả các vấn đề nghiên cứu lẫn thực tiễn, các thuật toán nghiên cứu sinh đề xuất được áp dụng cho các bộ dữ liệu UCI dataset hoặc NCI dataset như Balance scale, Kr-vs-kp, Breast cancer, Car, Tic-tac-toe, Molecula, HIV AIDS, Chemical compound, ... trong một số kết quả thử nghiệm. Các bộ dữ liệu trên dành cho nghiên cứu là các bộ dữ liệu đã được làm sạch, chuyển đổi phù hợp với các phương pháp khai phá dữ liệu trong các công trình khoa học. Để ứng dụng được vào thực tiễn [87], [96], cần phải thực hiện các công đoạn làm sạch dữ liệu, biến đổi dữ liệu trước khi áp dụng các thuật toán khai phá dữ liệu trong luận án này.

## 4. CẤU TRÚC LUẬN ÁN

Cấu trúc luận án có 3 chương như sau:

- Chương 1. Kiến thức chuẩn bị: Chương này trình bày một số các định nghĩa cơ sở, các định lý của các lý thuyết sẽ được áp dụng vào các phương pháp phát triển các thuật toán trong luận án này như lý thuyết tập thô, lý thuyết cơ sở dữ liệu quan hệ, lý thuyết đồ thị, lý thuyết phân tích khái niệm chính thức, lý thuyết về độ tin cậy, lý thuyết Dempster-Shafer.

- Chương 2. Chương này trình bày chi tiết về một số phương pháp nghiên cứu sinh đề xuất trong việc phát triển các thuật toán khai phá dữ liệu trên biểu diễn dữ liệu có cấu trúc dạng bảng như rút gọn đối tượng trong thời gian đa thức, rút

gọn thuộc tính không heuristic trong thời gian đa thức và sinh cây quyết định với thời gian thực hiện nhanh hơn thuật toán ID3, đồng thời nghiên cứu sinh cũng chứng minh tính đúng đắn và đầy đủ của các phương pháp này.

- Chương 3. Chương này trình bày một số phương pháp nghiên cứu sinh đề xuất về khai phá dữ liệu trên biểu diễn dữ liệu cấu trúc dạng đồ thị như bài toán khai phá đồ thị con thường xuyên đóng và phân loại đồ thị đa nhãn theo lý thuyết Dempster-Shafer. Trong bài toán khai phá đồ thị con thường xuyên đóng, nghiên cứu sinh đề xuất phương pháp xác định đẳng cấu đồ thị con trong thời gian đa thức và trong bài toán phân loại đa nhãn đồ thị, nghiên cứu sinh đề xuất độ đo khoảng cách trên dàn giao khái niệm phục vụ cho quá trình phân loại, đồng thời nghiên cứu sinh cũng chứng minh tính đúng đắn và đầy đủ của các phương pháp này.

# CHƯƠNG 1

# KIẾN THỨC CHUẨN BỊ

## 1.1 Lý thuyết cơ sở dữ liệu quan hệ

Phần này trình bày một số định nghĩa trong cơ sở dữ liệu quan hệ. Kết hợp với các định nghĩa của lý thuyết tập thô, các định nghĩa về tập bằng nhau, hệ bằng nhau cực đại, khóa, phản khóa góp phần thực hiện nhiệm vụ rút gọn thuộc tính và rút gọn đối tượng trên bảng quyết định nhất quán.

[20] Cho $\Omega = \{a_1, ..., a_n\}$ là một tập hữu hạn không rỗng các thuộc tính. Với mỗi thuộc tính $a_i$ có một tập không rỗng $D(a_i)$ các giá trị có thể của thuộc tính đó. Một tập con hữu hạn của tích Đề các $D(a_1) \times D(a_2) \times ... \times D(a_n)$ được gọi là một quan hệ trên $\Omega$. Rõ ràng, một quan hệ trên $\Omega$ là tập các ánh xạ

$$h : \Omega \to \bigcup_{a \in \Omega} D(a),$$

với $h(a) \in D(a)$ với mọi $a \in \Omega$.

**Định nghĩa 1.1.1.** [21] Cho $R = \{h_1, ..., h_m\}$ là một quan hệ trên tập hữu hạn $\Omega$ của các thuộc tính và $A, B \subseteq \Omega$. Ta nói rằng $B$ *phụ thuộc hàm* vào $A$ trong $R$ (ký hiệu là $A \to B$) nếu và chỉ nếu:

$$(\forall h_i, h_j \in R)((\forall a \in A)(h_i(a) = h_j(a)) \Rightarrow (\forall b \in B)(h_i(b) = h_j(b))).$$

với $1 \leqslant i, j \leqslant m$.

Cho $F_R = \{(A, B) : A, B \subset \Omega, A \to B\}$ được gọi là một *họ đầy đủ* các phụ thuộc hàm trong $R$

**Định nghĩa 1.1.2.** [21] Cho $\Omega$ là một tập hữu hạn, và ký hiệu $P(\Omega)$ là tập chứa tất cả các tập con của tập thuộc tính của $\Omega$. Cho $F \subseteq P(\Omega) \times P(\Omega)$. Ta nói rằng $F$ là một *họ f* trên $\Omega$ nếu và chỉ nếu với mọi $A, B, C, D \subseteq \Omega$

$$1)(A, A) \in F.$$
$$2)(A, B) \in F, (B, C) \in F \Rightarrow (A, C) \in F.$$
$$3)(A, B) \in F, A \subseteq C, D \subseteq B \Rightarrow (C, D) \in F.$$
$$4)(A, B) \in F, (C, D) \in F \Rightarrow (A \cup C, B \cup D) \in F.$$

Rõ ràng $F_R$ là một họ f trên $\Omega$. Cũng có thể nói nếu $F$ là một họ f trên $\Omega$, thì có một quan hệ $R$ sao cho $F_R = F$. Ta ký hiệu $F^+$ là tập chứa tất cả các phụ thuộc hàm có thể suy ra được từ $F$ bởi các luật từ $1) - 4)$.

**Định nghĩa 1.1.3.** [21] Một *lược đồ quan hệ* $S$ là một cặp $\langle \Omega, F \rangle$, với $\Omega$ là một tập của các thuộc tính và $F$ là một tập các phụ thuộc hàm trên $\Omega$. Cho $F^+$ là một tập tất cả các phụ thuộc hàm có thể suy diễn được từ $F$ bởi các luật trong định nghĩa 1.1.2. Ký hiệu $A^+ = \{a : A \rightarrow \{a\} \in F^+\}$ được gọi là *đóng* của $A$ trên $S$. Rõ ràng rằng $A \rightarrow B \in F^+$ nếu và chỉ nếu $B \subseteq A^+$.

Rõ ràng, nếu $S = \langle \Omega, F \rangle$ là một lược đồ quan hệ, thì sẽ có một quan hệ $R$ trên $\Omega$ sao cho $F_R = F^+$. Một quan hệ như vậy được gọi là một quan hệ Armstrong của $S$.

$A_R^+ = \{a : A \rightarrow \{a\} \in F^+\}$ được gọi là đóng của $A$ trên quan hệ $R$.

**Định nghĩa 1.1.4.** [21] Cho $R$ là một quan hệ, $S = \langle \Omega, F \rangle$ là một lược đồ quan hệ và $A \subseteq \Omega$. $F$ là một họ f trên $\Omega$ và $A \subseteq \Omega$. Thì $A$ là một *khóa* của $R$ (một khóa của $S$) nếu $A \rightarrow \Omega$ ($A \rightarrow \Omega \in F^+$, $(A, \Omega) \in F$). $A$ là một *khóa tối tiểu* của $R$ $(S, F)$ nếu $A$ là một khóa của $R$ $(S, F)$ và bất kỳ tập con thực sự của $A$ không phải là khóa của $R$ $(S, F)$. Ký hiệu $K_R$ $(K_S, K_F)$ là tập tất cả các khóa tối tiểu của $R$ $(S, F)$.

Một họ $K \subseteq P(R)$ là một *hệ Sperner* trên $R$ nếu cho bất kỳ $A, B \in K$ kéo theo $A \nsubseteq B$.

Rõ ràng $K_R$ ($K_S, K_F$) là các hệ Sperner.

**Định nghĩa 1.1.5.** [21] Cho $K$ là một hệ Sperner trên $\Omega$. Ta định nghĩa tập *các phản khóa* of $K$, ký hiệu $K^{-1}$, như sau:

$$K^{-1} = \{A \subset \Omega : (B \in K) \Rightarrow (B \nsubseteq A) \; and \; (A \subset C) \Rightarrow (\exists B \in K)(B \subseteq C)\}.$$

Dễ dàng thấy rằng $K^{-1}$ cũng là một hệ Sperner trên $\Omega$.

Biết rằng nếu $K$ là một hệ Sperner bất kỳ, thì có một lược đồ quan hệ $S$ sao cho $K_S = K$.

Giả sử rằng nếu một hệ Sperner đóng vai trò quan trọng trong tập các khóa tối tiểu (các phản khóa), thì hệ Sperner đó không rỗng (không chứa $\Omega$). Ta xem xét sự so sánh của hai thuộc tính như bước cơ bản của các thuật toán. Theo đó, nếu giả sử rằng các tập con của $\Omega$ được biểu diễn như các danh sách được sắp xếp của các thuộc tính, thì một toán tử nhị phân trên hai tập con của $\Omega$ yêu cầu nhiều nhất $|\Omega|$ các bước cơ bản.

**Định nghĩa 1.1.6.** [21] Cho $R$ là một quan hệ trên $\Omega$ và $E_R$ là *tập bằng nhau* của $\Omega$, ví dụ, $E_R = \{E_{ij} : 1 \leqslant i < j \leqslant |R|\}$, mà $E_{ij} = \{a \in \Omega : h_i(a) = h_j(a)\}$. Cho $M_R = \{A \in P(\Omega) : \exists E_{ij} = A, \nexists E_{pq} : A \subset E_{pq}\}$. Thì $M_R$ được gọi là *hệ bằng nhau cực đại* của $\Omega$.

**Định nghĩa 1.1.7.** [21] Cho $S = \langle \Omega, F \rangle$ là một lược đồ quan hệ, $a \in \Omega$. Ký hiệu $K_a^S = \{A \subseteq \Omega : A \to \{a\}, \nexists B : (B \to \{a\})(B \subset A)\}$. $K_a^S$ được gọi là họ của các tập tối tiểu của thuộc tính $a$ trên $S$.

Rõ ràng, $\Omega \notin K_a^S$, $\{a\} \in K_a^S$ và $K_a^S$ là một hệ Sperner trên $\Omega$.

Tương tự tập $K_a^R = \{A \subseteq \Omega : A \to \{a\}, \nexists B \subseteq \Omega : (B \to \{a\})(B \subset A)\}$ được gọi là một họ các tập tối tiểu của thuộc tính $a$ trên $R$.

Nếu $K$ là một hệ Sperner trên $\Omega$ cũng như họ các tập tối tiểu của thuộc tính $a$ trên $R$ (hoặc $S$); nghĩa là $K = K^R$ (hoặc $K = K^S$), thì $K^{-1} = \left(K_a^R\right)^{-1}$ (hoặc $K^{-1} = \left(K_a^S\right)^{-1}$) là họ các tập con cực đại của $\Omega$ mà không là họ các tập tối tiểu của thuộc tính $a$, được xác định:

$$\left(K_a^R\right)^{-1} = \{A \subseteq \Omega : A \to \{a\} \notin F_R^+, A \subset B \Rightarrow B \to \{a\} \in F_R^+\},$$
$$\left(K_a^S\right)^{-1} = \{A \subseteq \Omega : A \to \{a\} \notin F^+, A \subset B \Rightarrow B \to \{a\} \in F^+\}.$$

Rõ ràng $\Omega \notin K_a^S$, $\Omega \notin K_a^R$, $\{a\} \in K_a^S$, $\{a\} \in K_a^R$ and $K_a^S$, $K_a^R$, $\left(K_a^S\right)^{-1}$, $\left(K_a^R\right)^{-1}$ là các hệ Sperner trên $\Omega$.

## 1.2 Lý thuyết tập thô

Phần này trình bày một số khái niệm cơ bản về lý thuyết tập thô như bảng thông tin, bảng quyết định, bảng quyết định nhất quán, quan hệ bất khả phân biệt, phân hoạch, lớp tương đương, rút gọn thuộc tính, ma trận phân biệt, tập lõi. Các định nghĩa này được áp dụng trong bài toán tìm một rút gọn thuộc tính không heuristic trong thời gian đa thức, tìm rút gọn đối tượng trong thời gian đa thức và xây dựng cây quyết định từ bảng quyết định nhất quán thu gọn cả hai chiều ngang và dọc dựa trên rút gọn thuộc tính và rút gọn đối tượng.

[65] Một *hệ thông tin* $S$ bộ bốn có thứ tự $S = (U, A, V, f)$ mà $U$ là một tập hữu hạn không rỗng *các đối tượng*, được gọi là *tập vũ trụ*; $A$ là một tập hữu hạn không rỗng *các thuộc tính*; $V = \bigcup_{a \in A} V_a$ và $V_a$ là *miền giá trị* của các thuộc tính $a$; $f : U \times A \to V$ là một hàm toàn thể, mà $f(x, a) \in V_a$ với mọi $a \in A$ và $x \in U$ được gọi là *hàm thông tin*. Hàm $f_x : A \to V$ mà $f_x(a) = f(x, a)$ với mọi $a \in A$ và $x \in U$ sẽ được gọi là *thông tin về* $x$ trong $S$. Ký hiệu $a(x) = f_x(a)$. Nếu $B = \{b_1, b_2, ..., b_k\} \subseteq A$ là tập con

các thuộc tính, thì tập $b_i(x)$ được ký hiệu như $B(x)$. Theo đó, nếu $x, y$ là hai đối tượng trong $U$, thì $B(x) = B(y)$ nếu và chỉ nếu $b_i(x) = b_i(y), \forall i = 1, ..., k$.

**Định nghĩa 1.2.1.** [65] *Bảng quyết định* là hệ thông tin $S = (U, A, V, f)$, mà $A = C \cup D$ và $C \cap D = \varnothing$. Không mất tính tổng quát, giả sử $D$ chỉ chứa một thuộc tính quyết định $d$. Theo đó, từ đây xem bảng quyết định $DS = (U, C \cup \{d\}, V, f)$, mà $\{d\} \notin C$.

Cho bảng quyết định $DS = (U, C \cup \{d\}, V, f)$, có thể xem $U = \{u_1, ..., u_m\}$ là một quan hệ trên $C \cup \{d\}$.

Từ khái niệm của Pawlak [64] về sự phụ thuộc của luật quyết định từ tập thuộc tính điều kiện và thuộc tính quyết định và khái niệm phụ thuộc hàm trong cơ sở dữ liệu quan hệ [21] ta có định nghĩa sau.

**Định nghĩa 1.2.2.** [48] Một bảng quyết định $DS$ là *nhất quán* nếu và chỉ nếu phụ thuộc hàm $C \to \{d\}$ là đúng; nghĩa là cho bất kỳ $x, y \in U$ if $C(x) = C(y)$ thì $d(x) = d(y)$. Ngược lại, $DS$ là không nhất quán.

Theo định nghĩa tập bằng nhau và hệ bằng nhau cực đại 1.1.6 của lý thuyết cơ sở dữ liệu quan hệ, xem xét tập đối tượng $U$ của bảng quyết định nhất quán $DS = (U, C \cup \{d\}, V, f)$ là một quan hệ trên $C \cup \{d\}$, ta định nghĩa hệ bằng nhau cực đại đối với thuộc tính quyết định $d$ như sau.

**Định nghĩa 1.2.3.** [48] Cho $U = \{u_1, ..., u_m\}$ là một quan hệ trên $C \cup \{d\}$ của bảng quyết định nhất quán $DS$, $E_U$ tập bằng nhau của $U$. Đặt

$$M_d = \{A \in E_U : d \notin A, \nexists B \in E_U : d \notin B, A \subset B\}$$

được gọi là *hệ bằng nhau cực đại* của $U$ đối với thuộc tính quyết định $d$ của bảng quyết định nhất quán $DS$.

Hệ bằng nhau cực đại $M_d$ của $U$ đối với thuộc tính quyết định $d$ có ý nghĩa quan trọng trong các thuật toán của nghiên cứu sinh tìm rút gọn đối tượng và tìm một rút gọn thuộc tính bảng quyết định nhất quán $DS$.

**Định nghĩa 1.2.4.** [64] Mọi tập con các thuộc tính $P \subseteq C \cup D$ định ra một *quan hệ bất khả phân biệt*
$IND(P) = \{(u, v) \in U \times U | \forall a \in P, f(u, a) = f(v, a)\}$
$IND(P)$ định ra một *phân hoạch* trên $U$ được xác định bởi $U/P$.
Bất kỳ thành phần $[u]_P = \{v \in U | (u, v) \in IND(P)\}$ trong $U/P$ được gọi là một *lớp tương đương*.

Quan hệ bất khả phân biệt, phân hoạch và lớp tương đương được sử dụng trong quá trình tìm tất cả các tập rút gọn thuộc tính của một bảng quyết định nhất quán trong lý thuyết tập thô.

**Định nghĩa 1.2.5.** [64] xác định xấp xỉ trên, xấp xỉ dưới và miền dương dựa trên lớp tương đương như sau:

- *B-xấp xỉ trên* của $X$ là tập $\overline{B}X = \{u \in U | [u]_B \cap X \neq \varnothing\}$,
- *B-xấp xỉ dưới* của $X$ là tập $\underline{B}X = \{u \in U | [u]_B \subseteq X\}$ với $B \subseteq C$, $X \subseteq U$,
- *B-vùng biên* là tập $BN_B(X) = \overline{B}X \backslash \underline{B}X$,
- *B-miền dương* của $D$ là tập $POS_B(D) = \bigcup_{X \in U/D} (\underline{B}X)$

Xấp xỉ trên, xấp xỉ dưới, vùng biên, miền dương là các khái niệm quan trọng trong lý thuyết tập thô. Dựa trên khái niệm miền dương, Pawlak [64] định nghĩa rút gọn thuộc tính (có thể gọi tắt là rút gọn) để phân biệt với khái niệm rút gọn đối tượng do nghiên cứu sinh đề xuất.

**Định nghĩa 1.2.6.** [64] Cho $DS = (U, C \cup \{d\}, V, f)$ là một bảng quyết định. Nếu $B \subseteq C$ thỏa mãn:

$$1) POS_B(D) = POS_C(D)$$

$$2) \forall b \in B, POS_{B-\{b\}}(D) \neq POS_C(D)$$

thì $B$ được gọi là *rút gọn thuộc tính* của $C$.

Nếu $DS$ là một bảng quyết định nhất quán, $B$ là một *rút gọn thuộc tính* của $C$ nếu $B$ thỏa mãn $B \to \{d\}$ và $\forall B' \subset B$, $B' \nrightarrow \{d\}$. Cho $RED(C)$ là tập tất cả các rút tọn của $C$. Từ định nghĩa 1.2.6 và công thức $K_a^R$ trong định nghĩa 1.1.7 ta có $RED(C) = K_d^U - \{d\}$ với $K_d^U$ là họ tất cả các tập tối tiểu của thuộc tính $\{d\}$ của quan hệ $U$ trên tập thuộc tính $C \cup \{d\}$.

**Định nghĩa 1.2.7.** [64] Cho $U = \{u_1, u_2, ..., u_m\}$ là tập vũ trụ trên một bảng quyết định $DS$. *Ma trận phân biệt* được xác định bởi:

$$m_{ij} = \{a \in C : (a(u_i) \neq a(u_j)) \wedge (d(u_i) \neq d(u_j)), d \in D, \forall i, j = 1, 2, ..., m\}$$

với $m_{ij}$ là tập tất cả các thuộc tính mà phân loại các đối tượng $u_i$ và $u_j$ vào trong các lớp quyết định khác nhau trong phân hoạch $U/D$.

Theo Pawlak, định nghĩa ma trận phân biệt được sử dụng cho nhiều mục đích. Trong luận án này, nghiên cứu sinh chỉ áp dụng ma trận phân biệt trong việc xây dựng tập $CORE$. Tập $CORE$ là tập chứa mọi thuộc tính mà góp mặt trong mọi rút gọn thuộc tính trên miền dương theo lý thuyết tập thô của Pawlak.

**Định nghĩa 1.2.8.** [64] Tập tất cả các thuộc tính mà tham gia vào mọi rút gọn của $C$ được gọi là tập lõi của $C$ ký hiệu $CORE(C)$. Vì vậy $CORE(C) = \cap RED(C)$ với $RED(C)$ là tập tất cả các rút gọn của $C$. Tập lõi là tập chứa tất cả các thành phần đơn (tập chỉ có một phần tử) của ma trận phân biệt. Do vậy,

$$CORE(C) = \{a \in C : (\exists i, j) m_{ij} = \{a\}\}$$

Tập $CORE(C)$ được nghiên cứu sinh sử dụng làm nền tảng đề xuất xây dựng cây quyết định được từ bảng quyết định nhất quán thu gọn theo cả hai chiều ngang và dọc, dựa trên kết quả tìm một rút gọn đối tượng và một út gọn thuộc tính trong thời gian đa thức.

## 1.3 Lý thuyết đồ thị

Phần này, nghiên cứu sinh trình bày một số định nghĩa về đồ thị phục vụ cho thuật toán khai phá đồ thị con thường xuyên đóng và giải quyết bài toán con của nó là xác định đẳng cấu đồ thị con trong thời gian đa thức với ràng buộc về sử dụng máy truy cập ngẫu nhiên, tính có thứ tự của tập nhãn của đỉnh và cạnh để xây dựng đồ thị nhãn chuẩn hóa.

**Định nghĩa 1.3.1.** [44] Một *đồ thị được gắn nhãn* $G = (V, E, \sum_V, \sum_E, l)$ với $V$ là tập đỉnh, $E \subseteq V \times V$ là tập cạnh. $\sum_V$ và $\sum_E$ tương ứng là tập nhãn của đỉnh và nhãn của cạnh. Hàm gắn nhãn $l$ xác định ánh xạ $V \to \sum_V$ và $E \to \sum_E$.

**Định nghĩa 1.3.2.** [44] Không mất tính tổng quát, giả sử có một thứ tự tổng quát $\geqslant$ trên tập nhãn $\sum_V$ và $\sum_E$. Một đồ thị $G = (V, E, \sum_V, \sum_E, l)$ là một *đồ thị con* của đồ thị khác $G' = (V', E', \sum'_V, \sum'_E, l')$ nếu và chỉ nếu:
(i) $V \subseteq V'$,
(ii) $\forall u \in V, (l(u) = l'(u))$,
(iii) $E \subseteq E'$ và
(iv) $\forall (u, v) \in E, (l(u, v) = l'(u, v))$.
$G'$ cũng được gọi là *đồ thị cha* của $G$.

**Định nghĩa 1.3.3.** [44] Hai đồ thị $G = (V, E, \sum_V, \sum_E, l)$, $G' = (V', E', \sum'_V, \sum'_E, l')$ là *đẳng cấu* nếu và chỉ nếu tồn tại một song ánh $f : V \to V'$ sao cho:
(i) $\forall u \in V, (l(u) = l'(f(u)))$,
(ii) $\forall u, v \in V, ((u, v) \in E) \Leftrightarrow (f(u), f(v)) \in E'$,

(iii) $\forall (u,v) \in E, (l(u,v) = l'(f(u), f(v))$.

**Định nghĩa 1.3.4.** [44] Một đồ thị gắn nhãn $G$ là *đồ thị con đẳng cấu* với một đồ thị gắn nhãn $G'$, ký hiệu $G \subseteq G'$, nếu và chỉ nếu tồn tại một đồ thị con $G$" của $G'$ sao cho $G$ là đẳng cấu với $G$"

**Định nghĩa 1.3.5.** [44] Cho một tập đồ thị $\mathbb{GD}$ (cũng được gọi là cơ sở dữ liệu đồ thị) và một *ngưỡng* độ hỗ trợ tối thiểu $\sigma (0 \leqslant \sigma \leqslant 1)$, độ hỗ trợ của một đồ thị $G$, ký hiệu $sup_G$ được định nghĩa là phân số giữa số lượng các đồ thị $G' \in \mathbb{GD}$ và tổng số lượng các đồ thị của $\mathbb{GD}$ mà $G$ là đồ thị con đẳng cấu của $G'$:

$$sup_G = \frac{\mid \{G' \in \mathbb{GD} : G \subseteq G'\} \mid}{\mid \mathbb{GD} \mid}$$

$G$ là *thường xuyên* nếu và chỉ nếu $sup_G \geqslant \sigma$. Vấn đề khai phá đồ thị con thường xuyên là cho một ngưỡng $\sigma$ và một cơ sở dữ liệu đồ thị $\mathbb{GD}$, tìm tất cả *các đồ thị con thường xuyên* trong $\mathbb{GD}$.

Theo [89], nếu $g$ là một *đồ thị con* của $g'$, thì $g'$ là một *đồ thị cha* của $g$, ký hiệu $g \subseteq g'$ (*đồ thị cha đúng* nếu $g \subset g'$). Tập các *đồ thị con thường xuyên* của $\mathbb{GD}$, ký hiệu là $FS$, chứa tất cả các đồ thị con có độ hỗ trợ không ít hơn ngưỡng độ hỗ trợ tối thiểu, $\sigma$. Tập các *đồ thị con thường xuyên đóng* của $\mathbb{GD}$, $CS$, được định nghĩa là $CS = \{(g' \mid g' \in FS) \wedge \big(\nexists g \in FS : (g \subset g') \ \wedge \ (sup_g = sup_{g'})\big)\}$.

**Định nghĩa 1.3.6.** [44] Cho một $n \times n$ ma trận kề $M$ của một đồ thị $G$ với $n$ đỉnh, định nghĩa mã của $M$ , ký hiệu $code(M)$, có dạng chuỗi bằng cách ghép các thành phần thấp của ma trận tam giác của $M$ (gồm cả các thành phần trong đường chéo) theo thứ tự: $m_{1,1}m_{2,1}m_{2,2}...m_{n,1}m_{n,2}...m_{n,n-1}m_{n,n}$ mà $m_{i,j}$ là thành phần ở hàng thứ $i$ và cột thứ $j$ trong $M$ $(0 < j \leqslant i \leqslant n)$. Giả sử các hàng trong $M$ được đánh số từ 1 tới $n$ từ trên xuống dưới và các cột được đánh số từ 1 tới $n$ từ trái sang phải.

Trong [44], tác giả dùng thứ tự từ điển chuẩn (standard lexicographic order) theo trình tự để xác định một thứ tự toàn thể của hai mã bất kỳ $p$ và $q$. Cho một đồ thị

$G$, *dạng chuẩn* (canonical form) của nó là mã cực đại trong tất cả các mã có thể. Ma trận kề $M$ xuất ra dạng chuẩn được định nghĩa là *ma trận dạng chuẩn* của $G$ ($G$'s canonical adjacency matrix, CAM của $G$). Vậy $code(CAM(G))$ là mã của ma trận dạng chuẩn của đồ thị $G$.

## 1.4 Tập có thứ tự và dàn giao (lattices)

Phần này nghiên cứu sinh trình bày một số định nghĩa tập có thứ tự, dàn giao, tập đóng, hệ đóng. Dựa trên ánh xạ giữa tập đóng, hệ đóng với đồ thị con thường xuyên đóng để xây dựng dàn giao khái niệm từ đó xây dựng độ đo khoảng cách giữa các phần tử trên dàn giao khái niệm. Độ đo khoảng cách trên dàn giao khái niệm được ứng dụng trong bài toán phân loại đồ thị đa nhãn.

**Định nghĩa 1.4.1.** [19] Cho một quan hệ hai ngôi $\leqslant$ trên tập $P$, $(P, \leqslant)$ thỏa mãn ba điều kiện (*phản xạ*, *phản đối xứng* và *bắc cầu*) được gọi là một *tập có thứ tự* (hay *tập thứ tự bộ phận* hay *poset*). Tập tất cả các tập $P(X)$, gồm tất cả các tập con của $X$, nếu bao gồm thứ tự: cho $A, B \in P(X)$, xác định $A \leqslant B$ nếu và chỉ nếu $A \subseteq B$.

**Định nghĩa 1.4.2.** [32] Cho $Y \subseteq X$, với $(X, \leqslant)$ là một poset. Một toán tử *meet* hay *infimum* (hay *inf*) trên các tập con của tập $X$ ký hiệu $m = inf(Y)$ nếu và chỉ nếu: (i) $\forall y \in Y : m \leqslant y$, và (ii) $\forall m' \in X : (\forall y \in Y : m' \leqslant y) \Rightarrow m' \leqslant m$. $m$ cũng được gọi là *lớn nhất cận dưới* ($glb$) của tập $Y$. Một toán tử *join* hay *supremum* (hay *sup*) $s = sup(Y)$ nếu và chỉ nếu: (i) $\forall y \in Y : y \leqslant s$, và (ii) $\forall s' \in X : (\forall y \in Y : y \leqslant s') \Rightarrow s \leqslant s'$. $s$ cũng được gọi là *nhỏ nhất cận trên* ($lub$) của tập $Y$. Ký hiệu *glb* của $\{a, b\}$ bởi $a \wedge b$, và *lub* của $\{a, b\}$ bởi $a \vee b$. Lớn nhất cận dưới và nhỏ nhất cận trên không phải lúc nào cũng tồn tại với một poset.

**Định nghĩa 1.4.3.** [32] Một poset $(X, \leqslant)$ là một *dàn giao* nếu và chỉ nếu $\forall x, y \in X$ : $x \vee y$ và $x \wedge y$ cùng tồn tại.

**Định nghĩa 1.4.4.** [32] Một *khoảng* $[x, y]$ trong một poset $(X, \leqslant)$ được xác định: $\{z \mid x \leqslant z \leqslant y\}$. Một poset là *hữu hạn cục bộ* nếu tất cả các khoảng là hữu hạn.

**Định nghĩa 1.4.5.** [61] Một *hệ thống tập hợp* trên một tập $S$ là một họ $\Psi$ các tập con của $S$. Một *hệ đóng* $\zeta$ trên một tập $S$ là một hệ thống tập hợp trên $S$ thỏa mãn hai thuộc tính sau: (i) $S \in \zeta$ và (ii) $C_1, C_2 \in \zeta \Rightarrow C_1 \cap C_2 \in \zeta$. Các tập của hệ đóng $\zeta$ được gọi là *các tập đóng* của $\zeta$.

**Định lý 1.4.6.** [61] Một hệ đóng $(\zeta, \subseteq)$ là một dàn giao $(\zeta, \subseteq, \wedge, \vee)$ với (i) $C_1 \wedge C_2 = C_1 \cap C_2$ và $C_1 \vee C_2 = \cap\{C \in \zeta : C_1 \cup C_2 \subseteq C\}$.

**Định lý 1.4.7.** [61] Bất kỳ dàn giao nào cũng đẳng cấu với dàn giao của tập đóng của hệ đóng.

## 1.5 Phân tích khái niệm chính thức (FCA)

Phần này trình bày một số định nghĩa về ngữ cảnh chính thức, khái niệm chính thức, mối quan hệ cha - con giữa các khái niệm chính thức và dàn giao khái niệm. Từ những khái niệm này kết hợp với lý thuyết đồ thị, nghiên cứu sinh đề xuất xây dựng độ đo tương tự giữa hai đồ thị trên dàn giao khái niệm phục vụ giải quyết bài toán phân loại đa nhãn trên đồ thị.

**Định nghĩa 1.5.1.** [30] Một *ngữ cảnh chính thức* là một bộ ba $(G, M, I)$, với $G$ là một tập các đối tượng, $M$ là một tập các thuộc tính và $I$ là một quan hệ giữa $G$ và $M$, $I \subseteq G \times M$. Một kết nối Galois giữa $G$ và $M$ được xác định như sau:

$$A^I = \{m \in M | \forall g \in A, (g, m) \in I\}, A \subseteq G \tag{1.1}$$

$$B^I = \{g \in G | \forall m \in B, (g, m) \in I\}, B \subseteq M \tag{1.2}$$

**Định nghĩa 1.5.2.** [30] Một *khái niệm chính thức* là một cặp $(A, B)$, với $A \subseteq G$ là một tập con các đối tượng, $B \subseteq M$ là một tập con các thuộc tính, mà các đẳng thức

(1.1) $A^I = B$ và $A = B^I$ (1.2), với $A$ được gọi là *phạm vi* của khái niệm, và $B$ được gọi là *ý định* của khái niệm.

**Định nghĩa 1.5.3.** [30] Các khái niệm (theo định nghĩa 1.5.2) của một ngữ cảnh cho trước có một thứ tự mặc định *quan hệ khái niệm con - khái niệm cha* được xác định bởi

$$((A_1, B_1) \leqslant (A_2, B_2)) \Leftrightarrow A_1 \subseteq A_2 \ (\Leftrightarrow B_2 \subseteq B_1). \tag{1.3}$$

Tập có thứ tự của tất cả các khái niệm chính thức của $(G, M, I)$ được ký hiệu $\mathfrak{B}(G, M, I)$ và được gọi là *dàn giao khái niệm* của $(G, M, I)$ theo định nghĩa 1.4.2, 1.4.3.

**Định lý 1.5.4.** (*Định lý cơ bản về dàn giao khái niệm*) [30] Dàn giao khái niệm (theo định nghĩa 1.5.3) $\mathfrak{B}(G, M, I)$ là *dàn giao đầy đủ* với infimum và supremum được cho bởi:

$$\bigwedge_{t \in T}(A_t, B_t) = \left(\bigcap_{t \in T} A_t, \left(\bigcup_{t \in T} B_t\right)^{II}\right) \tag{1.4}$$

$$\bigvee_{t \in T}(A_t, B_t) = \left(\left(\bigcup_{t \in T} A_t\right)^{II}, \bigcap_{t \in T} B_t\right) \tag{1.5}$$

Định lý này thể hiện rằng đối với dàn giao khái niệm bất kỳ đều là dàn giao đầy đủ, nghĩa là mọi cặp phần tử đều có infimum và supremum. Áp dụng cho dàn giao khái niệm được xây dựng từ tập đồ thị con thường xuyên đóng, mọi cặp đồ thị con thường xuyên đóng đều có phần tử infimum và supremum. Theo đó, có thể xác định được độ đo khoảng cách giữa hai đồ thị con trên dàn giao khái niệm thông qua các phần tử infimum và supremum của nó.

## 1.6 Biến đổi và đồng biến đổi Mobius

Biến đổi và đồng biến đổi Mobius được nghiên cứu sinh sử dụng trong việc xây dựng các hàm như hàm cấp phát khối, hàm niềm tin theo lý thuyết độ tin cậy

Dempster-Shafer, từ mối quan hệ trên dàn giao khái niệm của các đồ thị, để phục vụ bài toán phân loại đa nhãn đồ thị sử dụng lý thuyết Dempster-Shafer.

**Định nghĩa 1.6.1.** [36] Cho $(L, \leqslant)$ là một poset hữu hạn cục bộ có một phần tử ở đáy dàn giao. Bất kỳ hàm $f$ trên $(L, \leqslant)$, *biến đổi Möbius* của $f$ là hàm $m : L \to \mathbb{R}$ giải pháp của phương trình

$$f(x) = \sum_{y \leqslant x} m(y). \tag{1.6}$$

Phương trình này luôn có một nghiệm duy nhất, một biểu thức của $m$ nhận được thông qua hàm Möbius $\mu : L^2 \to \mathbb{R}$ bởi

$$m(x) = \sum_{y \leqslant x} \mu(y, x) f(y) \tag{1.7}$$

với $\mu$ được xác định theo quy nạp bởi

$$\mu(x, y) = \begin{cases} 1, & if\ x = y \\ -\sum\limits_{x \leqslant t \leqslant y} \mu(x, t), & if\ x \leqslant y \\ 0 & otherwise. \end{cases} \tag{1.8}$$

**Định nghĩa 1.6.2.** [36] *Đồng biến đổi Möbius* của $f$, ký hiệu $q$, được xác định bởi

$$q(x) = \sum_{y \geqslant x} m(y),\ x \in L. \tag{1.9}$$

và $m$ có thể được khôi phục từ $q$ như sau:

$$m(x) = \sum_{y \geqslant x} \mu(x, y) q(y). \tag{1.10}$$

## 1.7 Lý thuyết Dempster-Shafer

Phần này trình bày một số định nghĩa cơ bản của lý thuyết Dempster-Shafer. Áp dụng luật Dempster trong việc kết hợp các hàm cấp phát khối và các hàm niềm tin

thông qua tập k láng giềng gần nhất của tập đồ thị con thường xuyên đóng dựa trên độ đo dàn giao khái niệm, từ đó xác định tập nhãn cho một đồ thị mới trong giải quyết bài toán phân loại đa nhãn cho đồ thị sử dụng lý thuyết Dempster-Shafer.

**Định nghĩa 1.7.1.** [36] Cho $\Omega$ là một không gian hữu hạn. Một hàm $m : 2^{\Omega} \rightarrow [0, 1]$ được gọi là một *hàm cấp phát khối* (hay đơn giản là *mass*) nếu $m(\varnothing) = 0$ and $\sum_{A \subseteq \Omega} m(A) = 1$. Một tập con $A \subseteq N$ được gọi là một *phần tử tiêu điểm* nếu $m(A) > 0$.

**Định nghĩa 1.7.2.** [36] Một *hàm niềm tin* trên $\Omega$ là một hàm $bel : 2^{\Omega} \rightarrow [0, 1]$ được sinh bởi một hàm cấp phát khối như sau:

$$bel(A) = \sum_{B \subseteq A} m(B),\ A \subseteq \Omega. \tag{1.11}$$

Chú ý rằng $bel(\varnothing) = 0$ và $bel(\Omega) = 1$. Hàm niềm tin nhận $m$ như biến đổi Möbius của $bel$, công thức nghịch đảo, nhận được bằng cách sử dụng các phương trình (1.7) và (1.8) là

$$m(A) = \sum_{B \subseteq A} (-1)^{|A \setminus B|} bel(B). \tag{1.12}$$

**Định nghĩa 1.7.3.** [36] Cho một cấp phát khối $m$, *hàm sự thật* được xác định bởi

$$pl(A) = \sum_{B | A \cap B \neq \varnothing} m(B) = 1 - bel(A^c),\ A \subseteq \Omega. \tag{1.13}$$

**Định nghĩa 1.7.4.** [36] Cho một cấp phát khối $m$, *hàm tính chất chung*, đồng biến đổi Möbius của $bel$ từ phương trình (1.9), được xác định bởi

$$q(A) = \sum_{A \subseteq B} m(B),\ A \subseteq \Omega \tag{1.14}$$

**Định nghĩa 1.7.5.** [36] Một *sức chứa* trên $\Omega$ là một hàm tập hợp $v : 2^{\Omega} \rightarrow [0, 1]$ mà $v(\varnothing) = 0, v(\Omega) = 1$, và $A \subseteq B$ kéo theo $v(A) \leqslant v(B)$ (*tính đơn điệu*). Các hàm sự thật và hàm niềm tin là các sức chứa. Cho bất kỳ một sức chứa $v$, *liên hợp* của nó được

xác định bởi $\bar{v}(A) = 1 - v(A^c)$. Do đó, các hàm sự thật là liên hợp của các hàm niềm tin (và ngược lại). Một sức chứa là *k-đơn điệu* ($k \geqslant 2$) nếu cho bất kỳ họ của $k$ các tập con $A_1, ..., A_k$ của $\Omega$ nó có:

$$v\left(\bigcup_{i\in K} A_i\right) \geqslant \sum_{I\subseteq K, I\neq\varnothing} (-1)^{|I|+1} v\left(\bigcap_{i\in I} A_i\right). \tag{1.15}$$

[74] chỉ ra rằng một sức chứa là tổng đơn điệu nếu và chỉ nếu nó là một hàm niềm tin, do đó tồn tại vài cấp phát khối sinh ra nó.

**Định nghĩa 1.7.6.** [36] Cho hai cấp phát khối $m_1, m_2$, *luật kết hợp Dempster* tính sự kết hợp của hai khối vào một khối:

$$m(A) = (m_1 \oplus m_2)(A) = \sum_{B_1\cap B_2 = A} m_1(B_1)m_2(B_2),\ \forall A \subseteq \Omega, A \neq \varnothing, \tag{1.16}$$

và $m(\varnothing) = 0$. Luật kết hợp Dempster có thể được tính thông qua các hàm tính chất chung, gọi $q, q_1, q_2$ là hàm tính chất chung kết hợp $m, m_1, m_2$, trở thành một đó là:

$$q(A) = q_1(A)q_2(A),\ \forall A \subseteq \Omega. \tag{1.17}$$

# CHƯƠNG 2

# KHAI PHÁ DỮ LIỆU DẠNG BẢNG

Trong chương này, nghiên cứu sinh trình bày các thuật toán đề xuất là các kết quả mới của luận án về tìm rút gọn đối tượng vẫn bảo toàn việc tìm tập tất cả các tập rút gọn thuộc tính trong thời gian đa thức, tìm một rút gọn thuộc tính không heuristic trong thời gian đa thức và xây dựng cây quyết định từ bảng thu gọn theo chiều dọc của bảng quyết định nhất quán trong thời gian đa thức.

Nội dung của chương này dựa trên các công trình số [2], [4], [6] trong danh mục công trình công bố của nghiên cứu sinh.

## 2.1 Đặt vấn đề

Trong hầu hết các hệ thống thông tin hiện nay sử dụng dữ liệu dạng bảng và áp dụng rất nhiều các kỹ thuật khác nhau để thực hiện khám phá tri thức từ dữ liệu. Lý thuyết tập thô được Zdzislaw Pawlak giới thiệu trong những năm 1980s [64] là một công cụ toán học giải quyết các vấn đề mơ hồ và không chắc chắn. Từ đó lý thuyết tập thô [67] đã được tìm thấy trong nhiều ứng dụng thú vị của trí tuệ nhân tạo và khoa học nhận dạng đặc biệt trong lĩnh vực học máy, sự thu nhận kiến thức, phân tích quyết định, khám phá tri thức từ các cơ sở dữ liệu, các hệ chuyên gia, các hệ thống hỗ trợ quyết định, lập luận suy dẫn, và nhận dạng mẫu. Lý thuyết này đặc biệt quan trọng trong việc áp dụng cho các *hệ thống thông tin* (thỉnh thoảng còn được gọi là các bảng dữ liệu, các bảng quyết định, các hệ thống thuộc tính giá trị hay các bảng hành động có điều kiện) trong các vấn đề biểu diễn tri thức, rút gọn thông tin, suy luận phụ thuộc và nhiều vấn đề khác.

Thông tin [66] về thế giới thực được cho trong dạng bảng thông tin (còn được gọi

là bảng quyết định). Bảng thông tin biểu diễn dữ liệu đầu vào, thu thập được từ bất kỳ miền nào, chẳng hạn như y học, tài chính hoặc quân sự. Các hàng của một bảng quyết định được gọi là các đối tượng. Các thuộc tính, các cột của bảng quyết định, của các đối tượng được gán giá trị theo một số biến. Có hai loại biến phân biệt: *thuộc tính* (còn được gọi là thuộc tính điều kiện), *quyết định* (còn được gọi là thuộc tính quyết định). Thông thường các hệ thống chỉ đòi hỏi các quyết định đơn. Ví dụ nếu bảng thông tin mô tả một bệnh viện, các đối tượng là các bệnh nhân; các thuộc tính là triệu chứng và xét nghiệm; quyết định là bệnh tật. Mỗi bệnh nhân được đặc trưng hóa bởi các kết quả xét nghiệm và các triệu chứng, và được phân loại mức độ nghiêm trọng của bệnh bởi các bác sĩ.

Khái niệm cơ bản của lý thuyết tập thô [66] là một quan hệ bất khả phân biệt kết hợp với một tập các thuộc tính. Rõ ràng quan hệ bất khả phân biệt là một quan hệ tương đương. Các thuộc tính dư thừa là các thuộc tính được xác định theo khái niệm của quan hệ bất khả phân biệt. Nếu một tập thuộc tính và tập cha của nó có cùng một quan hệ bất khả phân biệt thì bất kỳ thuộc tính nào thuộc vào tập cha mà không thuộc tập con đó là thuộc tính dư thừa. Một tập các thuộc tính mà không chứa thuộc tính dư thừa được gọi là tối thiểu. Tập hợp $P$ của các thuộc tính được gọi là một rút gọn của một tập $Q$ các thuộc tính nếu $P$ là tối thiểu và quan hệ bất khả phân biệt được xác định trên $P$ và trên $Q$ là giống nhau.

Bảng thông tin trong dữ liệu lớn rất khó khai phá do tính lớn của dữ liệu dẫn đến nhu cầu giảm bớt dữ liệu trước khi khai phá dữ liệu. Đa phần các phương pháp làm giảm dữ liệu bảng dữ liệu lớn thuộc về giảm dữ liệu theo chiều dọc, nghĩa là giảm đi số thuộc tính của bảng (giảm số cột của bảng) hay còn gọi là rút gọn thuộc tính [59], [92]. Bảng được nói đến trong các công trình này là bảng thông tin [66] theo lý thuyết tập thô [64]. Đã có nhiều công trình nghiên cứu về thực hiện rút gọn thuộc tính trên bảng thông tin [59], [69], [100]. Một số khái niệm về rút gọn khác được đưa ra như rút gọn tương đối [60], rút gọn dựa trên miền dương [64], rút gọn dựa trên entropy

thông tin của Shannon [64], [69], rút gọn dựa trên mô hình tập thô cổ điển [68], [76], rút gọn dựa trên ma trận và hàm phân biệt [93], rút gọn dựa trên tìm kiếm heuristic [100], rút gọn dựa trên ba độ đo khoảng cách [60], rút gọn phân phối cận dưới $\beta$ và rút gọn phân phối cận trên $\beta$ theo khái niệm tập thô chính xác biến [59], rút gọn tri thức trong các hệ không nhất quán [51], [93], rút gọn thuộc tính sử dụng tập thô mờ và hạt thông tin [42], [92], rút gọn dựa trên độ đo mức quan trọng của thuộc tính trên mô hình tập thô mờ [42], rút gọn dựa trên chiến lược trong lựa chọn tập con thuộc tính như vỏ bọc và lọc tính toán hạt bằng cách dùng hạt thông tin từ số lượng đặc trưng và sự chuyển dạng số lượng thuộc tính trong các biến ngôn ngữ mờ [42], rút gọn thuộc tính theo định nghĩa của dàn giao khái niệm [12], [17], [56].

Trong [75] đánh giá độ phức tạp tính toán về rút gọn thuộc tính bảng quyết định áp dụng lý thuyết tập thô, rút gọn tối thiểu và phụ thuộc tối thiểu là vấn đề NP-hard. Trong [48] tìm tất cả các rút gọn thuộc tính dựa trên miền dương theo lý thuyết tập thô của Pawlak có độ phức tạp hàm mũ. Do độ phức tạp tính toán thời gian không đa thức của việc tìm tất cả các rút gọn thuộc tính nên phần lớn các thuật toán được đề cập trong các công trình trên phải sử dụng tiếp cận heuristic để tìm một rút gọn thuộc tính trong thời gian đa thức. Tìm một rút gọn thuộc tính được áp dụng trong thực tế về sinh cây quyết định hoặc sinh luật quyết định. Tối ưu nhất là tìm được rút gọn tối thiểu, nghĩa là với số thuộc tính ít nhất trong bảng có thể sinh ra tập luật đầy đủ để phân loại các đối tượng theo thuộc tính quyết định cho trước. Độ phức tạp tính toán của các thuật toán rút gọn thuộc tính bảng quyết định không chỉ phụ thuộc vào số lượng thuộc tính điều kiện mà còn phụ thuộc vào số lượng đối tượng của bảng quyết định. Dữ liệu lớn chính là vấn đề về số lượng đối tượng ngày càng tăng và làm cho quá trình tìm các rút gọn thuộc tính kể cả các thuật toán rút gọn thuộc tính càng ngày càng chậm ngay cả khi sử dụng tiếp cận heuristic. Do vậy, vấn đề đặt ra là cần phải tìm các rút gọn đối tượng sao cho số lượng đối tượng được thu nhỏ lại trong khi vẫn bảo toàn các thuật toán tìm rút gọn thuộc tính.

## 2.2 Loại bỏ thuộc tính dư thừa

Bảng quyết định thường chứa các đối tượng không nhất quán là các đối tượng bằng nhau trên tập thuộc tính điều kiện nhưng khác nhau trên tập thuộc tính quyết định và được gọi là bảng quyết định không nhất quán. Tuy nhiên, phụ thuộc vào từng lớp vấn đề cần giải quyết, một bảng quyết định không nhất quán có thể chuyển đổi về bảng quyết định nhất quán bằng cách loại bỏ các đối tượng không nhất quán. Trong một bảng quyết định bất kỳ, nếu không cho phép hai hàng hoàn toàn trùng nhau, có thể dễ dàng kiểm tra bảng quyết định nhất quán hay không nhất quán bằng một thuật toán có độ phức tạp tính toán thời gian đa thức đối với kích thước của bảng đó.

Hơn nữa bảng quyết định nhất quán cũng thường chứa các thuộc tính dư thừa. Sự hiện diện của các thuộc tính này làm độ phức tạp tính toán trong vấn đề khai phá dữ liệu tăng lên đáng kể. Các thuộc tính dư thừa là các thuộc tính không xuất hiện trong bất kỳ một rút gọn nào. Việc loại bỏ các thuộc tính dư thừa trước khi tiến hành khai phá dữ liệu trên bảng quyết định nhất quán có ý nghĩa thực tiễn cao trong ngữ cảnh mà dữ liệu ngày càng gia tăng, đa dạng và phức tạp. Vấn đề tìm tập tất cả các thuộc tính dư thừa tương đương với vấn đề tìm tất cả các thuộc tính rút gọn. Để giải quyết vấn đề này, phương pháp tiếp cận thông thường là tìm họ tất cả các rút gọn của bảng quyết định nhất quán rồi tìm hợp của tất cả các tập rút gọn. Tuy nhiên phương pháp này là không khả thi cho các bảng dữ liệu lớn trong khía cạnh độ phức tạp tính toán của thuật toán tìm họ tất cả các rút gọn của bảng quyết định nhất quán là hàm mũ đối với tập thuộc tính điều kiện [48].

Sự kết hợp của việc loại bỏ các thuộc tính dư thừa với rút gọn đối tượng của bảng quyết định nhất quán sẽ cho ra kết quả là một bảng quyết định nhất quán không dư thừa và là bước làm sạch trong quá trình khai phá dữ liệu. Chẳng hạn như đối với bảng quyết định nhất quán với thuộc tính quyết định là có đi chơi Golf hay không (bảng 2.1) với bộ tổ hợp các điều kiện với quang cảnh là nắng, mưa, trời quang; nhiệt độ

cao, thấp, trung bình; trời có gió mạnh hay nhẹ; độ ẩm cao, thấp, trung bình; cỏ ẩm hay khô; số lỗ gôn nhiều hay ít; bảng này có kích thước là 6 thuộc tính và có 14 đối tượng, sau quá trình loại bỏ thuộc tính dư thừa còn lại 4 thuộc tính (bảng 2.2) và sau quá trình rút gọn đối tượng sẽ chỉ còn lại 6 đối tượng (bảng 2.3). Như vậy, bảng quyết định nhất quán không dư thừa sẽ nhỏ hơn nhiều bảng quyết định nhất quán gốc ban đầu và vẫn đảm bảo tìm ra được tất cả cá rút gọn thuộc tính của bảng gốc, kích thước bảng nhỏ hơn đồng nghĩa với vấn đề lưu trữ bảng để xử lý sau này cũng làm giảm không gian lưu trữ đặc biệt trong bối cảnh dữ liệu lớn đang là xu hướng thời đại, mọi dữ liệu đều có thể được thu thập, lưu trữ và xử lý.

Để loại bỏ các thuộc tính dư thừa bảng quyết định nhất quán, phần này trình bày một số các hàm, thủ tục và thuật toán như sau:

- Cho bảng quyết định nhất quán $DS = (U, C \cup \{d\}, V, f)$ là một quan hệ trên $C \cup \{d\}$
- *EqualitySet* để tìm tập bằng nhau trong bảng quyết định nhất quán,
- *MaximalEqualityForD* để tìm hệ bằng nhau cực đại của thuộc tính quyết định,
- Định nghĩa tập các thuộc tính không dư thừa *REAT(C)*
- *RemoveRedundantAttribute* để loại bỏ các thuộc tính dư thừa trong bảng quyết định nhất quán.
- Định lý chứng tỏ *REAT(C)* là đúng đắn
- Bổ đề $M_d = (K_d^r)^{-1}$ được sử dụng trong việc tìm rút gọn đối tượng bảng quyết định nhất quán.

**Procedure** EqualitySet($DS$)

**Đầu vào:** $DS = (U, C \cup \{d\}, V, f)$, $POS_C(\{d\}) = U$, $C = \{c_1, ..., c_n\}$, $U = \{u_1, ..., u_m\}$

**Đầu ra :** $E_r$

Xem xét quan hệ $r = \{u_1, ..., u_m\}$ trên tập thuộc tính $R = C \cup \{d\}$.;
$E_r \leftarrow \varnothing$;
**foreach** *cặp* $(i \in U, j \in U) \wedge (i < j)$ **do**
    **if** $\forall A_k \in R : f(u_i, A_k) == f(u_j, A_k)$ **then**
        $(E_{i,j} \leftarrow A_k)$;
    **end**
    $E_r \leftarrow \{E_{i,j}\}$;
**end**
trả về $E_r$;

**Procedure** MaximalEqualityForD($E_r$)

**Đầu vào:** $E_r$

**Đầu ra :** $M_d$

$M_d \leftarrow \varnothing$;
**foreach** $i = 0; i < |E_r|; i++$ **do**
    $M_d \leftarrow E_r[i] \notin \{d\}$;
    **if** $\exists A, B \in M_d : A \subseteq B$ **then**
        Loại bỏ $A$ ra khỏi $M_d$;
    **end**
**end**
trả về $M_d$;

**Định lý 2.2.1.** [20] Cho $K$ là một hệ Sperner trên $\Omega$. Thì

$$\bigcup_{A \in K} A = \Omega - \bigcap_{B \in K^{-1}} B$$

**Định nghĩa 2.2.2.** [80] Cho một bảng quyết định nhất quán $DS = (U, C \cup \{d\}, V, f)$ với $U = \{u_1, ..., u_m\}$ trên tập thuộc tính $R = C \cup \{d\}$, từ định nghĩa 1.2.6 thì $RED(C) = K_d^r - \{d\}$, nếu ký hiệu $REAT(C)$ tập tất cả các thuộc tính rút gọn của $C$ thì:

$$REAT(C) = \bigcup_{A \in RED(C)} A = \left( \bigcup_{A \in K_d^r} A \right) - \{d\}$$

**Algorithm 1:** RemoveRedundantAttribute($DS$)

**Đầu vào:** $DS = (U, C \cup \{d\}, V, f)$, $POS_C(\{d\}) = U$, $C = \{c_1, ..., c_n\}$, $U = \{u_1, ..., u_m\}$

**Đầu ra :** $REAT(C)$

1 $E_r \leftarrow EqualitySet(DS)$;

2 $M_d \leftarrow MaximalEqualityForD(E_r)$;

3 $N = (C \cup \{d\}) - \bigcap_{B \in M_d} B$;

4 trả về $REAT(C) = N - \{d\}$;

**Định lý 2.2.3.** [80] $REAT(C)$ là tập chứa tất cả các thuộc tính rút gọn của $C$.

Có thể thấy rằng số bước tính $E_r$ (thủ tục *EqualitySet($DS$)*) không lớn hơn $|R| * |U|^2$ và số bước tính $M_d$ (thủ tục *MaximalEqualityForD($E_r$)*) không lớn hơn $|E_r|^2$. Do vậy, độ phức tạp tính toán tồi nhất của thuật toán *RemoveRedundantAttribute($DS$)* là $O(|U|^4 + |C \cup \{d\}|)$ đa thức theo hàng và cột của $DS$.

Bổ đề 2.2.4 được nghiên cứu sinh sử dụng trong thuật toán tìm một rút gọn đối tượng bảng quyết định nhất quán trong thời gian đa thức.

**Bổ đề 2.2.4.** [79] Cho bảng quyết định nhất quán $DS = (U, C \cup \{d\}, V, f)$ với $C = \{c_1, c_2, ..., c_n\}, U = \{u_1, u_2, ..., u_m\}$. Xem $DS$ như một quan hệ $r = \{u_1, u_2, ..., u_m\}$ trên tập thuộc tính $R = C \cup \{d\}$.

Đặt $E_r = \{E_{ij} : 1 \leqslant i < j \leqslant m\}$ với $E_{ij} = \{a \in R : a(u_i) = a(u_j)\}$.

Đặt $M_d = \{A \in E_r : d \notin A, \nexists B \in E_r : d \notin B, A \subset B\}$.
Thì $M_d = (K_d^r)^{-1}$ với $K_d^r$ là họ các thuộc tính tối tiểu của thuộc tính $\{d\}$ trên quan hệ $r$.

## 2.3 Rút gọn thuộc tính không heuristic

Tìm các rút gọn từ bảng quyết định là một trong các mục tiêu chính trong xử lý thông tin. Nhiều nghiên cứu tập trung vào rút gọn thuộc tính tức là làm giảm số cột trong bảng quyết định [59], [92]. Thật không may là tìm tất cả các rút gọn thuộc tính trong một bảng quyết định là vấn đề có độ phức tạp hàm mũ [48]. [75] đã chứng minh rằng vấn đề tìm rút gọn thuộc tính tối thiểu (một rút gọn có số lượng thuộc tính là ít nhất) hay phụ thuộc tối thiểu là vấn đề NP-hard. Trong hầu hết các công trình tìm rút gọn thuộc tính bảng thông tin quyết định nhất quán, để tìm rút gọn thuộc tính tối thiểu trước hết phải tìm tất cả các rút gọn thuộc tính sau đó chọn ra một rút gọn thuộc tính có số thuộc tính ít nhất gọi là rút gọn thuộc tính tối thiểu. Hay nói một cách khác thì các công trình tìm rút gọn thuộc tính theo miền dương của bảng quyết định nhất quán sẽ phải đi tìm tất cả các miền dương của tất cả các tổ hợp của bộ thuộc tính của bảng rồi mới tìm được rút gọn thuộc tính của các tổ hợp và quyết định rút gọn thuộc tính tổ hợp nào là nhỏ nhất. Như vậy, kể cả khi đi tìm một rút gọn thuộc tính tức là đi tìm một tổ hợp các thuộc tính là rút gọn của tập tất cả các thuộc tính của bảng thông tin quyết định nhất quán là một hàm của $2^n$ với $n$ là số thuộc tính của bảng. Như vậy, thời gian tính toán có độ phức tạp thuộc lớp không đa thức NP.

Một số công trình công bố sử dụng tìm một rút gọn thuộc tính bảng quyết định sử dụng phương pháp heuristic. Trong luận án này, nghiên cứu sinh đề xuất một phương pháp đi tìm một rút gọn thuộc tính không heuristic trong thời gian đa thức. Bằng cách kết hợp lý thuyết cơ sở dữ liệu quan hệ và lý thuyết tập thô, thuật toán rút gọn thuộc tính không sử dụng heuristic duyệt tuần tự từng thuộc tính điều kiện đối với thuộc tính

quyết định và loại trừ dần dần các thuộc tính theo hệ bằng nhau cực đại cho thuộc tính quyết định. Thuật toán *AnAttributeReduct* nghiên cứu sinh đề xuất tìm một rút gọn thuộc tính được chứng minh tính đúng đắn và thực hiện trong thời gian đa thức.

Phần này trình bày một số hàm và thủ tục về khóa, phản khóa trong lý thuyết cơ sở dữ liệu quan hệ, rút gọn thuộc tính miền dương trong lý thuyết tập thô và thuật toán nghiên cứu sinh đề xuất tìm một rút gọn thuộc tính không heuristic [2] trong thời gian đa thức như sau:

- *AntikeySet* tìm tập phản khóa trong một quan hệ. Theo định nghĩa (1.1.7) thì tìm được tập phản khóa của một quan hệ thì cũng tìm được tập khóa tối tiểu của quan hệ đó.
- *AMinimalKey* tìm một khóa tối tiểu của quan hệ $C \cup \{d\}$ dựa trên tập phản khóa tìm được bởi *AntikeySet*.
- *MinimalKeySet* tìm tập khóa tối tiểu của quan hệ trên $C \cup \{d\}$ dựa trên *Antikey-Set* và *AMinimalKey*.
- *AnAttributeReduct* tìm một rút gọn thuộc tính không heuristic [6] trong thời gian đa thức.

Theo định nghĩa (1.2.5) và (1.2.6) thì tập các tập tối tiểu của thuộc tính quyết định $\{d\}$ là tập tất cả các tập rút gọn thuộc tính của bảng quyết định nhất quán $DS$. Theo định nghĩa này thì $RED(C) = K_d^U - \{d\}$, nghĩa là tập rút gọn thuộc tính của bảng quyết định nhất quán $DS$ là tập tất cả các khóa tối tiểu của quan hệ trên $C \cup \{d\}$ đã loại bỏ thuộc tính $\{d\}$. Xây dựng trên tính chất này, nghiên cứu sinh đề xuất tìm một khóa tối tiểu của quan hệ trên $C \cup \{d\}$ mà không chứa thuộc tính quyết định $\{d\}$ dựa

trên thuật toán tìm một khóa tối tiểu *AMinimalKey* trong cơ sở dữ liệu quan hệ.

**Procedure** AntikeySet($\{B_1, ..., B_m\}$)

**Đầu vào:** Cho $K = \{B_1, ..., B_m\}$ là một hệ Sperner trên $U$

**Đầu ra :** $K^{-1}$

1 $K^{-1} \leftarrow \{U - \{a\} : a \in B_1\}$, $K_1 = \{B_1\}^{-1}$.;

2 **while** $q < m$ **do**

3 Giả sử $K_q = F_q \cup \{X_1, ..., X_{t_q}\}$, với $X_1$, ..., $X_{t_q}$ là thành phần của $K$ chứa $B_{q+1}$ và $F_q = \{A \in K_q : B_{q+1} \nsubseteq A\}$;

4 **foreach** $i \in [1, t_q]$ **do**

5 Xây dựng tập phản khóa của $\{B_{q+1}\}$ trên $X_i$ tương tự như với $K_1$, là tập cực đại của $X_i$ không chứa $B_{q+1}$. Ký hiệu $A^i_1$, ... , $A^i_{r_i}$;

6 $K_{q+1} = F_q \cup \{A^i_p : A \in F_q \Rightarrow A^i_p \nsubseteq A, 1 \leqslant i \leqslant t_q, 1 \leqslant p \leqslant r_i\}$;

7 **end**

8 **end**

9 trả về $K^{-1} = K_m$;

**Định lý 2.3.1.** [20] Cho mọi $q$ ($1 \leqslant q \leqslant m$), $K_q = \{B_1, ..., B_q\}^{-1}$, v.v..., $K_m = K^{-1}$

[78] Cho $K_0 = U$, có $K_q = F_q \cup \{X_1, ..., X_t\}$, với $1 \leqslant q \leqslant m-1$. Ký hiệu $l_q$ là số phần tử của $K_q$. Khi xây dựng $K_{q+1}$, thời gian tồi nhất là $O(|U|^2(l_q - t_q))$ nếu $t_q < l_q$ và $O(|U|^2 t_q)$ nếu $l_q = t_q$. Tổng thời gian là $O\left(|U|^2 \sum_{q=0}^{m-1} t_q u_q\right)$ với $u_q = l_q - t_q$ nếu $l_q > t_q$ và $u_q = 1$ ngược lại.

Theo định nghĩa (1.1.7), có thể thấy rằng $K$ và $K^{-1}$ là duy nhất và quyết định lẫn nhau, mọi thành phần của $K$ không tồn tại trong $K^{-1}$ và ngược lại đồng thời cả $K$ và $K^{-1}$ đều là hệ Sperner, thêm vào đó $K^{-1}$ được xây dựng bởi *AntikeySet($K$)* không

phụ thuộc vào thứ tự của các thành phần $B_1, ..., B_m$.

**Procedure** AMinimalKey($K$)

**Đầu vào:** Cho $K$, $H$ là các hệ Sperner và $C = \{c_1, ..., c_n\} \subseteq U$ mà $H^{-1} = K$ và $\exists B \in K : B \subseteq C$

**Đầu ra :** $D \in H$

```
A = C;
foreach i ∈ [1, n] do
    if ∀B ∈ K : A(i) − {c_{i+1}} ⊈ B then
        A = A − c_{i+1};
    end
end
trả về D = A(n) (A(n) là A khi vòng lặp thực hiện đến i = n);
```

**Bổ đề 2.3.2.** [78] Nếu $K$ là một tập các phản khóa, thì $A(n) \in H$.

**Algorithm 2:** MinimalKeySet($K$)

**Đầu vào:** Cho $K = \{B_1, ..., B_k\}$ là một hệ Sperner trên $U$

**Đầu ra :** $H$ mà $H^{-1} = K$

```
A_1 = AMinimalKey({B_1, ..., B_k}); K(1) = A_1;
foreach i ∈ [1, k] do
    if ∃B ∈ K_i^{-1} : B ⊈ B_j (∀j : 1 ⩽ j ⩽ k) then
        A_{i+1} = AMinimalKey(B ∈ K_i^{-1}); K(i + 1) = K(i);
    else
        H = K(i);
    end
end
trả về H;
```

[78] độ phức tạp tính toán thời gian của *MinimalKeySet(K)* là

$$O\left(n\left(\sum_{q=1}^{m-1}(kl_q + nt_q u_q) + k^2 + n\right)\right),$$
với $|U| = n$, $|K| = k$, $|H| = m$, cũng nghĩa là $l_q$, $t_q$, $u_q$.

Theo [11], kích thước của một hệ Sperner trên $U$ không thể lớn hơn $C_n^{[n/2]}$, với $n = |U|$. $C_n^{[n/2]}$ là tiệm cận bằng $\frac{2^{\frac{n+1}{2}}}{\pi n^{\frac{1}{2}}}$. Từ đó, độ phức tạp tính toán thời gian tồi nhất của thuật toán *MinimalKeySet(K)* không thể nhiều hơn hàm mũ của số lượng các thuộc tính. Trong trường hợp $l_q < l_m$ ($q = 1, ..., m-1$), dễ dàng thấy được độ phức tạp tính toán thời gian của thuật toán *MinimalKeySet* không lớn hơn $O(|U|^2|K||K^{-1}|^2)$.

Thuật toán *MinimalKeySet(K)* dùng để tìm tập tất cả các tập rút gọn miền dương $RED(C)$ của bảng quyết định nhất quán $DS$ theo định nghĩa 1.2.6 mà không quan tâm đến các đối tượng trong bảng quyết định nhất quán. Phương pháp này thay thế cho phương pháp tìm $RED(C)$ của $DS$ theo phân hoạch các lớp tương đương.

**Algorithm 3:** AnAttributeReduct($DS$)

**Đầu vào:** $DS = (U, C \cup \{d\}, V, f)$

**Đầu ra :** $D \in RED(C)$

```
E_r ← EqualitySet(DS);
M_d ← MaximalEqualityForD(E_r);
C = {c_1, ..., c_n}, H = C;
foreach i = 0; i < n; i++ do
    if ∄B ∈ M_d : H − c_{i+1} ⊆ B then
        H = H − c_{i+1};
    end
end
trả về D = H(n) (H(n) là H khi vòng lặp kết thúc với i = n = |C|);
```

**Định lý 2.3.3.** $H(n) \in RED(C)$.

*Chứng minh.* Thuật toán *AnAttributeReduct($DS$)* [2] được xây dựng dựa trên thuật toán *AMinimalKey($K$)*. Theo bổ đề 2.2.4, $\left(K_d^r\right)^{-1} = M_d$. Theo bổ đề 2.3.2, $H(n) \in K_d^r$ (1). Theo kết quả của định nghĩa 1.2.6, $RED(C) = K_d^r - \{d\}$ (2). Ở bước 3 của thuật toán *AnAttributeReduct($DS$)* đặt $C = \{c_1, ..., c_n\}$ thì $d \notin C$. Do vậy, trong thuật toán *AnAttributeReduct($DS$)* $d \notin H(n)$ (3). Từ (1) và (3) có $H(n) \in K_d^r - \{d\}$ (4). Từ (2) và (4) đạt được $H(n) \in RED(C)$. Định lý được chứng minh. □

Độ phức tạp tính toán thời gian của thuật toán *AnAttributeReduct($DS$)* không lớn hơn $O(|C| \times |U|^4)$. Có thể thấy được rằng nếu thay đổi thứ tự các phần tử của tập $C$ ở bước 3, có thể nhận được một rút gọn thuộc tính khác từ bảng quyết định nhất quán $DS$. Như vậy, sẽ có $|C|!$ tổ hợp các thành phần của tập thuộc tính $C$. Vì thế, tồn tại một thuật toán có độ phức tạp thời gian không đa thức để tìm tất cả các rút gọn thuộc tính từ một bảng quyết định nhất quán. Tuy nhiên để tìm một rút gọn thuộc tính thì lại có độ phức tạp tính toán thời gian đa thức chẳng hạn như thuật toán *AnAttributeReduct* do nghiên cứu sinh đề xuất.

## 2.4 Rút gọn đối tượng bảng quyết định nhất quán

Trong bảng quyết định thì rút gọn thuộc tính đóng vai trò quan trọng trong khai phá dữ liệu, tuy nhiên số lượng các thuộc tính dư thừa và thuộc tính rút gọn không bị loại trừ nhiều so với bảng gốc. Dữ liệu càng tăng đồng nghĩa với số đối tượng ngày càng tăng và khi áp dụng rút gọn đối tượng thì số lượng đối tượng có thể bị loại trừ rất lớn. Chẳng hạn như với bảng quyết định về chơi Golf thì số thuộc tính là 4 (bảng 2.2), số đối tượng là 14 thì khi rút gọn thuộc tính chỉ có 1 thuộc tính bị loại trừ (bảng 2.4), trong khi đó có đến 8 đối tượng bị loại trừ khi áp dụng rút gọn đối tượng (bảng 2.3). Rút gọn đối tượng dựa trên lý thuyết tập thô [64], [75] và lý thuyết cơ sở dữ liệu quan hệ [20], [21], [78] mà nghiên cứu sinh đề xuất là một phương pháp rút gọn các đối tượng của bảng quyết định nhất quán mà vẫn bảo toàn được thông tin khi tìm các tập

rút gọn thuộc tính. Thuật toán *AnObjectReduct* nghiên cứu sinh đề xuất có ý nghĩa quan trọng với một số tính chất sau:

- Rút gọn đối tượng trong bảng thông tin quyết định nhất quán, loại trừ một số đối tượng khỏi bảng quyết định nhất quán sao cho với tập các đối tượng còn lại, vấn đề tìm toàn bộ các rút gọn thuộc tính không bị ảnh hưởng hay nói cách khác là được bảo toàn. Như vậy, không gian lưu trữ của bảng thông tin quyết định nhất quán không còn quá lớn trong bối cảnh dữ liệu lớn hiện nay và thời gian thực hiện các thuật toán về rút gọn thuộc tính cũng giảm đi đáng kể.

- Thuật toán rút gọn đối tượng bảng quyết định nhất quán là một thuật toán có độ phức tạp tính toán thời gian đa thức. Theo đó, sẽ không tốn nhiều thời gian để xây dựng được một bảng thông tin quyết định nhất quán thu gọn từ bảng thông tin quyết định nhất quán ban đầu.

- Trong một số trường hợp, rút gọn đối tượng bảng quyết định nhất quán có thể xác định được một bảng quyết định nhất quán là không có rút gọn thuộc tính chẳng hạn như khi rút gọn đối tượng mà tất cả các đối tượng bị loại trừ thì bảng quyết định nhất quán đó không thể tìm được bất kỳ một rút gọn thuộc tính nào.

- Trong một số trường hợp, bảng thông tin quyết định nhất quán đã thu gọn đối tượng có thể vẫn giữ được yếu tố chính xác khi xây dựng các tập luật quyết định hoặc cây quyết định.

Phương pháp đề xuất dựa trên sự kết hợp lý thuyết tập thô và lý thuyết cơ sở dữ liệu quan hệ trong đó đối với lý thuyết tập thô dựa trên định nghĩa $RED(C)$ còn đối với lý thuyết cơ sở dữ liệu quan hệ là dựa trên định nghĩa về khóa của quan hệ $K_d^r$. Định nghĩa $RED(C) = K_d^r - \{d\}$ có ý nghĩa rất quan trọng trong việc tìm rút gọn đối tượng [2], công thức này có nghĩa rằng tập tất cả các rút gọn thuộc tính của bảng

quyết định nhất quán bằng với tập tất cả các khóa tối tiểu của quan hệ $U$ trên tập thuộc tính $C \cup \{d\}$ của bảng quyết định nhất quán $DS = (U, C \cup \{d\}, V, f)$ trừ đi thuộc tính quyết định $\{d\}$. Mặt khác, nếu có tồn tại một tập tất các các khóa tối tiểu của một quan hệ cho trước thì cũng tồn tại tập tất cả các phản khóa tức là tập chứa tất cả các tập không phải là khóa lớn nhất của quan hệ này. Tập chứa tất cả các khóa tối tiểu và tập chứa tất cả các tập phản khóa xác định lẫn nhau nghĩa có tập này thì phải có tập kia và ngược lại cùng với đó là tập chứa tất cả các khóa tối tiểu là duy nhất và tập chứa tất cả các phản khóa cũng là duy nhất đối với một quan hệ. Trong lý thuyết cơ sở dữ liệu quan hệ [78] có một định nghĩa về hệ bằng nhau và hệ bằng nhau cực đại, có thể hiểu rằng nếu hai quan hệ có hệ bằng nhau cực đại như nhau thì hai quan hệ là như nhau trên tập phụ thuộc hàm nghĩa là tập phụ thuộc hàm bất biến cho dù hai quan hệ này có các dòng với các giá trị tại các thuộc tính là khác nhau. Dựa trên một bổ đề 2.2.4 đã được chứng minh $M_d = (K_d^r)^{-1}$. Nếu tạm thời lược bỏ một đối tượng của bảng quyết định nhất quán mà không $M_d$ không thay đổi có nghĩa là $(K_d^r)^{-1}$ cũng không thay đổi theo đó tất cả các khóa tối tiểu cũng không thay đổi và $RED(C)$ không thay đổi do đó có thể kết luận rằng đối tượng này là dư thừa có thể thực sự xóa bỏ khỏi bảng quyết định nhất quán và ngược lại nếu tạm thời lược bỏ đối tượng mà $M_d$ thay đổi thì đối tượng đó bắt buộc bị giữ lại. Cuối cùng thu được một rút gọn đối tượng của bảng quyết định nhất quán là một bảng quyết định nhất quán thu gọn theo hàng trong đó các đối tượng không ảnh hưởng đến quá trình tìm tất cả các rút gọn thuộc tính đã bị loại bỏ hoàn toàn.

**Định nghĩa 2.4.1.** Một rút gọn đối tượng của bảng quyết định nhất quán $DS = (U, C \cup \{d\}, V, f)$ là một bảng quyết định nhất quán $DS' = (U', C \cup \{d\}, V, f)$, với $RED(C) = RED_U(C)$ và:

1) $U' \subseteq U$,

2) $RED_U(C) = RED_{U'}(C)$,

3) $RED_U(C) \neq RED_{U'-\{u\}}(C)$, $\forall u \in U'$.

**Algorithm 4:** AnObjectReduct($DS$)

**Đầu vào:** $DS = (U, C \cup \{d\}, V, f)$

**Đầu ra :** $DS' = (U', C \cup \{d\}, V, f)$

```
E_r ← EqualitySet(DS);
M_d^U ← MaximalEqualityForD(E_r);
T = U = {u_1, ..., u_m};
foreach i = 0; i < |U|; i++ do
    if M_d^{T-u_{i+1}} = M_d^U then
        T = T - u_{i+1};
    end
end
trả về DS' = (U' = T(m), C ∪ {d}, V, f) (T(m) là T sau khi vòng lặp kết
    thúc với i = m = |U|);
```

**Định lý 2.4.2.** $DS' = (U' = T(m), C \cup \{d\}, V, f)$ thỏa mãn ba điều kiện 1), 2) và 3) theo định nghĩa 2.4.1.

*Chứng minh.* Chứng minh bằng quy nạp. Ở bước cơ bản $T(0) = U$, rõ ràng, $U' = U$, $RED_{U'}(C) = RED_U(C)$ theo đó hai điều kiện 1), 2) được thỏa. Bước quy nạp, giả sử $T(i) = U(i)$ thỏa mãn hai điều kiện 1), 2) theo định nghĩa 2.4.1. Cần phải chứng minh rằng $T(i+1) = U(i+1)$ cũng thỏa mãn hai điều kiện đó.

Trường hợp đầu tiên: Nếu $T(i+1) = T(i)$ thì rõ ràng là $U(i+1) = U(i)$, $RED_{U(i+1)}(C) = RED_{U(i)}(C) = RED(C)$ theo giả thiết quy nạp. Do vậy, $T(i+1)$ thỏa mãn hai điều kiện 1), 2) theo định nghĩa 2.4.1.

Trường hợp thứ hai: Nếu $T(i+1) = T(i) - \{u_{i+1}\}$ thì $M_d^U = M_d^{U(i+1)}$. Theo bổ đề 2.2.4, $M_d^U = \left(K_d^U\right)^{-1}$ với $(U = \{u_1, ..., u_m\}) \Rightarrow M_d^{U(i+1)} = \left(K_d^{U(i+1)}\right)^{-1} \Rightarrow \left(K_d^U\right)^{-1} = \left(K_d^{U(i+1)}\right)^{-1}$. Theo định nghĩa 1.1.5 và 1.1.7 ($K$ và $K^{-1}$ là duy nhất

quyết định lẫn nhau), có thể thấy rằng $\left(K_d^U\right) = \left(K_d^{U(i+1)}\right)$. Từ định nghĩa 1.2.6 và kết quả của định nghĩa 1.2.6, dẫn đến $RED_U(C) = \left(K_d^U\right) - \{d\}$ và $RED_{U(i+1)}(C) = \left(K_d^{U(i+1)}\right) - \{d\} \Rightarrow$ (ii1) $RED_U(C) = RED_{U(i+1)}(C)$. Từ giả thiết quy nạp, có (ii2) $RED_U(C) = RED_{U(i)}(C)$. Từ (ii1), (ii2) đạt được $RED_U(C) = RED_{U(i)}(C) = RED_{U(i+1)}(C)$. Vì $RED_U(C) = RED(C)$ là một hệ Sperner (theo định nghĩa $K_d^U$ là một hệ Sperner và $\Rightarrow K_d^U - \{d\}$ là một hệ Sperner), $RED_{U(i)}(C)$ và $RED_{U(i+1)}(C)$ là các hệ Sperner. Cuối cùng, hai điều kiện theo định nghĩa 2.4.1 được thỏa ở bước $i+1$ như sau:

1) $U(i+1) \subseteq U(i)$,

2) $RED_{U(i+1)}(C) = RED_{U(i)}(C) = ... = RED_U(C) = RED(C)$ Khi $i+1 = m$ thì thuật toán *AnObjectReduct($DS$)* dừng. Bây giờ cần phải chứng minh $U(m)$ thỏa mãn điều kiện 3) theo định nghĩa 2.4.1 nghĩa là $RED_{U(m)-u}(C) \neq RED_U(C)$ với $\forall u \in U(m)$. Giả sử rằng có tồn tại một $u = u_{i+1}$, $u \in U(m)$ mà $RED_{U(m)-u_{i+1}}(C) = RED_U(C)$ $(ii3)$. Theo định nghĩa 1.2.6, $RED_{U(m)-u_{i+1}}(C) = K_d^{U(m)-u_{i+1}} - \{d\}$ và $RED_U(C) = K_d^U - \{d\}$, do vậy

$$(ii3) \Leftrightarrow K_d^{U(m)-u_{i+1}} - \{d\} = K_d^U - \{d\} \Leftrightarrow K_d^{U(m)-u_{i+1}} = K_d^U (ii4)$$

Theo định nghĩa 1.1.5, 1.1.7 và bổ đề 2.2.4 ($K$ và $K^{-1}$ là duy nhất quyết định lẫn nhau), có nghĩa là

$$(ii4) \Leftrightarrow \left(K_d^{U(m)-u_{i+1}}\right)^{-1} = \left(K_d^U\right)^{-1} \Leftrightarrow M_d^{U(m)-u_{i+1}} = M_d^U (ii5)$$

Theo chứng minh quy nạp trên, nếu $M_d^{U(m)-u_{i+1}} = M_d^U$ thì $u_{i+1}$ sẽ phải bị loại bỏ, do đó $u_{i+1} \notin U(m)$ trái ngược với giả thiết $u = u_{i+1} \in U(m)$. Do vậy, điều kiện 3) theo định nghĩa 2.4.1 được thỏa mãn. Định lý được chứng minh. □

Rõ ràng số bước tính toán $E_r$ theo định nghĩa 1.1.6 là ít hơn $|U|^2$. Số bước tính toán $M_d$ là ít hơn $|E_r|^2$ và $|E_r| \leqslant \dfrac{|U|(|U|-1)}{2}$. Do vậy, độ phức tạp thời gian tồi nhất của thuật toán *AnObjectReduct($DS$)* không lớn hơn $O(|U|^5)$. Có thể dễ dàng thấy

rằng nếu thay đổi trật tự các phần tử của tập vũ trụ $U$, có thể tìm được một rút gọn đối tượng khác. Có $|U|!$ tổ hợp các bộ đối tượng của bảng quyết định nhất quán $DS$. Vì vậy, vấn đề tìm tất cả các rút gọn đối tượng của bảng quyết định nhất quán có độ phức tạp tính toán thời gian không đa thức. Tuy nhiên, tìm một rút gọn đối tượng của bảng quyết định nhất quán có độ phức tạp thời gian tồi nhất là đa thức theo như thuật toán *AnObjectReduct* nghiên cứu sinh đề xuất. Sự kết hợp của thuật toán *AnObjectReduct* và thuật toán *RemoveRedundantAttribute* sẽ cho ra một bảng quyết định nhất quán có đầy đủ các đặc tính của bảng quyết định gốc trong các vấn đề rút gọn thuộc tính hoặc trong một số trường hợp sinh luật quyết định và sinh cây quyết định.

## 2.5 Xây dựng cây quyết định từ bảng rút gọn

Cây quyết định là một đồ thị không có chu trình có một gốc, đỉnh trong là các thuộc tính hoặc giá trị các thuộc tính điều kiện và lá là các quyết định. Đường đi từ gốc đến lá là một luật quyết định được sinh từ bảng quyết định. Cây quyết định là một ngữ cảnh đơn giản nhất cho phép đưa ra một tập các luật quyết định một cách trong sáng và dễ hiểu nhất. Việc tìm luật quyết định dựa vào cây quyết định cũng nhanh hơn so với tìm luật quyết định trong một tập luật quyết định do cấu trúc cây hiệu quả hơn nhiều so với cấu trúc danh sách. Việc sinh cây quyết định tạo ra tập luật quyết định từ một bảng quyết định trong thời gian đa thức có ý nghĩa quan trọng trong khai phá dữ liệu đặc biệt là phân lớp dữ liệu cho các đối tượng của bảng quyết định.

Hầu hết các nghiên cứu xây dựng cây quyết định đều áp dụng heuristic với hàm lượng giá (information gain) như ID3, C4.5 để xây dựng cây quyết định nhằm hai mục đích. Mục đích thứ nhất là xây dựng cây quyết định từ bảng quyết định sao cho thu được nhiều luật cô đọng nhất nghĩa là đường đi từ gốc đến lá là ngắn nhất. Mục đích thứ hai là thuật toán sinh cây quyết định từ bảng quyết định là thuật toán có độ phức tạp tính toán thời gian là đa thức để đáp ứng kịp thời dữ liệu lớn ngày càng phát triển.

Trong luận án này, nghiên cứu sinh đề xuất một phương pháp sinh cây quyết định IRDT [4] không sử dụng hàm độ đo thông tin (Entropy và Information Gain) trong lý thuyết thông tin như các thuật toán ID3, C4.5 hoặc CART. Thuật toán sinh cây quyết định IRDT (Indiscernibility Relation Decision Tree) thay thế hàm entropy để lựa chọn các các thuộc tính trong quá trình sinh cây quyết định. Sự thay thế hàm Entropy với quan hệ bất khả phân biệt thể hiện đúng đắn bản chất của phân loại đúng đối tượng với lớp nằm trong thuộc tính quyết định của quan hệ bất khả phân biệt đó. Do vậy, thuật toán IRDT thực hiện phân loại (phân lớp) đối tượng tốt hơn ID3, C4.5 và cây quyết định được sinh ra từ IRDT có độ chính xác cao hơn ID3, C4.5. Cây quyết định được sinh theo IRDT không hoàn toàn trùng khớp với các cây quyết định do thuật toán ID3, C4.5 sinh, tuy nhiên vẫn đảm bảo sinh ra đầy đủ tập luật quyết định từ bảng quyết định. Hơn nữa, thuật toán IRDT cũng thực hiện nhanh hơn thuật toán ID3 trong quá trình sinh cây quyết định.

Ý tưởng thuật toán IRDT là thực hiện đệ quy việc tìm các thuộc tính tốt nhất để làm nút trong việc sinh cây quyết định. Nút tốt nhất là nút đạt tiêu chí có nhiều nhất số lượng tập trong quan hệ bất khả phân biệt của thuộc tính đó mà là tập con của tập bất khả phân biệt của thuộc tính quyết định $\{d\}$ của bảng quyết định nhất quán. Sau mỗi lần chọn được thuộc tính tốt nhất sẽ tìm được các giá trị của tập con của tập thuộc tính đó dựa trên quan hệ bất khả phân biệt của phân hoạch theo thuộc tính đó. Tương ứng với mỗi giá trị của các tập con theo quan hệ bất khả phân biệt sẽ gọi đệ quy với các thuộc tính còn lại của tập thuộc tính ban đầu đã loại đi thuộc tính tốt nhất. Kết quả cuối cùng sẽ được cây quyết định. Thủ tục *RecursiveNode* thực hiện tất cả các bước của ID3 và thay thế hàm tính độ đo thông tin (entropy, information gain) bằng giá trị *bestAttribute* là giá trị lớn nhất về số lượng của thuộc tính điều kiện đối với thuộc tính quyết định dựa trên quan hệ bất khả phân biệt của phân hoạch theo thuộc tính điều kiện. Thủ tục *RecursiveNode* trả về con trỏ trỏ về đỉnh gốc của cây. Thuật toán IRDT đơn giản chỉ cần gọi thủ tục *RecursiveNode* và được trả về đỉnh gốc của cây quyết

định.

**Procedure** RecursiveNode(DS)

1 $Node \leftarrow \varnothing$;
2 **if** $((|U| == 1)||(|C \cup \{d\}| == 0))$ **then**
3 $\quad Node \leftarrow U(d)$ (nút lá);
4 **else**
5 $\quad bestAttribute \leftarrow max\,(\forall(e \in C \cup \{d\}) \sum (IND(e) \subseteq IND(d)))$;
6 $\quad remainAttributes \leftarrow (C \cup \{d\} - bestAttribute)$;
7 $\quad Node \leftarrow bestAttribute$;
8 $\quad Node.childs \leftarrow \{RecursiveNode(DS')\}$;
9 $\quad (DS' = (U' = U : Value(bestAttribute =$
$\quad\quad v), (C \cup \{d\}) - bestAttribute, V, f))$, $(\forall v \in Value(bestAttribute))$;
10 **end**
11 trả về $Node$;

**Định lý 2.5.1.** Thủ tục $RecursiveNode(DS)$ là đúng đắn.

*Chứng minh.* Để chứng minh thủ tục $RecursiveNode(DS)$ là đúng đắn ta sẽ chứng minh dựa vào thuật toán ID3 là đúng đắn. Ở đây, chỉ thay thế việc tính giá trị lớn nhất của Gain cho thuộc tính tốt nhất trong thuật toán ID3 bằng việc tính tổng tất cả các tập con của thuộc tính tốt nhất thỏa mãn điều kiện tập con đó theo quan hệ bất khả phân biệt của thuộc tính tốt nhất phải là tập con đúng của tập con bất kỳ theo quan hệ bất khả phân biệt của thuộc tính quyết định $d$. Như vậy, ta nhận thấy thủ tục $RecursiveNode(DS)$ chỉ thay thế hàm tính độ đo thông tin (Information Gain) bằng $IND(e) \subseteq IND(d)$ trong bảng quyết định $DS$. Việc tính độ đo thông tin thể hiện tương quan giữa thuộc tính tốt nhất và thuộc tính quyết định trong khi đó $IND(e) \subseteq IND(d)$ cũng thể hiện tính tương quan giữa thuộc tính tốt nhất $e$ với thuộc tính quyết định $d$. Vậy nếu ID3 là thuật toán đúng đắn thì thủ tục $RecursiveNode(DS)$ cũng là

đúng đắn. Định lý được chứng minh. □

**Algorithm 5:** IRDT($DS$)

**Đầu vào:** $DS = (U, C \cup \{d\}, V, f)$

**Đầu ra :** $DecisionTree(DS)$

1 $Root \leftarrow RecursiveNode(DS = (U, C \cup \{d\}, V, f))$;

2 trả về $Root$;

**Định lý 2.5.2.** Thuật toán $IRDT(DS)$ là đúng đắn.

*Chứng minh.* Do thủ tục $RecursiveNode(DS)$ là đúng đắn nên thuật toán $IRDT(DS)$ là đúng đắn. Định lý được chứng minh. □

Vấn đề xây dựng tất cả các cây quyết định từ bảng quyết định từ một bảng quyết định $DS$ là một vấn đề NP-đầy đủ bởi sẽ có $|C|!$ sự sắp xếp các thuộc tính để tạo cây quyết định. Do vậy, xây dựng tất cả các cây quyết định từ bảng quyết định là không khả thi, nhất là trong bối cảnh dữ liệu lớn. Nghiên cứu sinh áp dụng phương pháp thu gọn bảng quyết định trước theo cả chiều ngang và chiều dọc bằng cách kết hợp hai thuật toán *AnObjectReduct($DS$)* và *AnAttributeReduct($DS$)* sau đó mới áp dụng tìm cây quyết định trên bảng thu gọn. Như vậy, thuật toán tìm cây quyết định trong thời gian đa thức với kích thước của bảng đã thu gọn sẽ nhanh hơn so bảng quyết định đầy đủ. Cây quyết định được sinh từ bảng quyết định thu gọn sẽ có số lượng luật quyết định ít hơn so với tập luật quyết định từ bảng quyết định đầy đủ. Nếu áp dụng thuật toán sinh cây quyết định sử dụng bảng quyết định thu gọn theo chiều ngang và chiều dọc đồng thời sử dụng tập $CORE(C)$ thì tốc độ tính toán nhanh hơn nhiều. Tập $CORE(C)$ chứa tất cả các thuộc tính đơn tham gia vào mọi tập rút gọn thuộc tính của bảng quyết định nhất quán $DS$. Như vậy rõ ràng, việc chọn một thuộc tính có nhiều giá trị nhất trong tập $CORE(C)$ làm đỉnh gốc của cây sẽ bớt đi công việc tính toán

cho các thuật toán sinh cây quyết định.

**Procedure** BuildCORE($DS$)

**Đầu vào:** $DS = (U, C \cup \{d\}, V, f)$

**Đầu ra :** CORE($C$) của $DS$

$CORE \leftarrow \varnothing$;

**foreach** $(i, j \in U) \wedge (i \neq j)$ **do**

  $CORE[i, j] \leftarrow A \subseteq C : f(u_i, A) = f(u_j, A)$

**end**

Loại bỏ các tập có nhiều hơn một thuộc tính khỏi $CORE$;

trả về $CORE$;

**Algorithm 6:** DecisionTree($DS$)

**Đầu vào:** $DS = (U, C \cup \{d\}, V, f)$

**Đầu ra :** Root

$DR \leftarrow RemoveRedundantAttribute(DS)$;

$DO \leftarrow AnObjectReduct(DR)$;

$DA \leftarrow AnAttributeReduct(DR)$;

$CORE(C) \leftarrow BuildCORE(DO)$;

$S \leftarrow x; x = \exists a \in CORE(C) : maxvalue(a)$;

$DB \leftarrow (U_{DO}, AnAttributeReduct(DR) \cup \{d\}, V, f)$;

$Root \leftarrow x$;

$Root.childs \leftarrow IRDT(DS' = (DO, (DA \cup \{d\}) - \{x\}, V, f))$;

## 2.6 Ví dụ thu gọn bảng và cây quyết định

Cho bảng quyết định nhất quán 2.1 $DS = (U, C \cup \{d\}, V, f)$ với $U = \{u_1, ..., u_{14}\}$ ($\{1, ..., 14\}$), $\{d\}$ là thuộc tính quyết định "Play Golf".

$C = \{Outlook, Grass, Temperature, Humidity, Windy, NumberHoles\}$

($\{o, g, t, h, w, n\}$ or $\{ogthwn\}$), $R = C \cup \{d\} = \{ogthwnd\}$,
$V_{Outlook} = \{Sunny, OverCast, Rain\}$, $V_{Temperature} = \{High, Middle, Low\}$,
$V_{Humidity} = \{High, Middle\}$, $V_{Grass} = \{Wet, Dry\}$,
$V_{Windy} = \{Weak, Strong\}$, $V_{NumberHoles} = \{20, 10\}$, $V_d = \{No, Yes\}$,
$V = V_{Outlook} \cup V_{Grass} \cup V_{Temperature} \cup V_{Humidity} \cup V_{Windy}$
$\cup V_{NumberHoles} \cup V_d$, và hàm $f : U \times C \cup \{d\} \rightarrow \bigcup_{a \in C} V_a$

Bảng 2.1: Bảng quyết định nhất quán gốc

| No. | Outlook | Grass | Temperature | Humidity | Windy | NumberHoles | d |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | Wet | High | High | Weak | 10 | No |
| 2 | Sunny | Dry | High | High | Strong | 20 | No |
| 3 | Overcast | Wet | High | High | Weak | 10 | Yes |
| 4 | Rain | Dry | Middle | High | Weak | 10 | Yes |
| 5 | Rain | Wet | Low | Middle | Weak | 20 | Yes |
| 6 | Rain | Wet | Low | Middle | Strong | 20 | No |
| 7 | Overcast | Dry | Middle | Middle | Strong | 20 | Yes |
| 8 | Sunny | Wet | Low | High | Weak | 10 | No |
| 9 | Sunny | Wet | Middle | Middle | Weak | 10 | Yes |
| 10 | Rain | Dry | Middle | Middle | Weak | 20 | Yes |
| 11 | Sunny | Dry | Middle | Middle | Strong | 20 | Yes |
| 12 | Overcast | Dry | Middle | High | Strong | 10 | Yes |
| 13 | Overcast | Dry | High | Middle | Weak | 20 | Yes |
| 14 | Rain | Dry | Middle | High | Strong | 10 | No |

Áp dụng thuật toán *RemoveRedundantAttribute($DS$)* cho bảng quyết định nhất quán gốc để loại bỏ toàn bộ các thuộc tính dư thừa (các thuộc tính không tham gia vào bất kỳ rút gọn nào). Đầu tiên ta tính $E_r$ chứa tất cả các $E_{i,j}$ như sau:
$E_{1,2} = othd$, $E_{1,3} = gthwn$, $E_{1,4} = hwn$, $E_{1,5} = gw$, $E_{1,6} = gd$, $E_{1,8} = oghwnd$,

$E_{1,9} = ogwn$, $E_{1,10} = w$, $E_{1,11} = o$, $E_{1,12} = hn$, $E_{1,13} = tw$, $E_{1,14} = hnd$, $E_{2,3} = th$, $E_{2,4} = gh$, $E_{2,5} = n$, $E_{2,6} = wnd$, $E_{2,7} = gwn$, $E_{2,8} = ohd$, $E_{2,10} = gn$, $E_{2,12} = ghw$, $E_{2,13} = gtn$, $E_{2,14} = ghwd$, $E_{3,4} = hwnd$, $E_{3,5} = gwd$, $E_{3,6} = g$, $E_{3,7} = od$, $E_{3,8} = ghwn$, $E_{3,9} = gwnd$, $E_{3,10} = wd$, $E_{3,11} = d$, $E_{3,12} = ohnd$, $E_{3,13} = otwd$, $E_{4,5} = owd$, $E_{4,7} = gtd$, $E_{4,9} = twnd$, $E_{4,10} = ogtwd$, $E_{4,12} = gthnd$, $E_{4,14} = ogthn$, $E_{5,8} = gtw$, $E_{5,10} = ohwnd$, $E_{6,10} = ohn$, $E_{7,9} = thd$, $E_{7,11} = gthwnd$, $E_{7,13} = oghnd$, $E_{9,10} = thwd$, $E_{9,12} = tnd$, $E_{9,13} = hwd$, $E_{9,14} = tn$, $E_{10,13} = ghwnd$, $E_{10,14} = ogt$, $E_{11,12} = gtwd$, $E_{11,13} = ghnd$, $E_{12,13} = ogd$

Tiếp theo, dựa trên $E_r$ đã được tính, ta tính $M_d$ từ đó tìm được $REAT(C)$

$M_d = \{gthwn, ogthn, ogwn\} = \{M_1, M_2, M_3\}$

$N = \bigcap_{M \in M_d} = M_1 \cap M_2 \cap M_3 = \{gthwn\} \cap \{ogthn\} \cap \{ogwn\} = \{gn\}$

$REAT(C) = N - \{d\} = \{gn\} - \{d\} = \{gn\}$

Theo $REAT(C)$ được tính bằng $\{gn\}$ thì hai thuộc tính "Grass" và "Number-Holes" là thuộc tính dư thừa (không tham gia bất kỳ rút gọn nào) không ảnh hưởng đến quá trình sinh luật quyết định nên loại bỏ hai thuộc tính này được bảng quyết định nhất quán *không dư thừa thuộc tính* (bảng 2.2) từ bảng quyết định nhất quán gốc 2.1.

Bảng 2.2: Bảng quyết định không dư thừa thuộc tính từ bảng gốc 2.1

| No. | Outlook | Temperature | Humidity | Windy | d |
|---|---|---|---|---|---|
| 1 | Sunny | High | High | Weak | No |
| 2 | Sunny | High | High | Strong | No |
| 3 | Overcast | High | High | Weak | Yes |
| 4 | Rain | Middle | High | Weak | Yes |
| 5 | Rain | Low | Middle | Weak | Yes |
| 6 | Rain | Low | Middle | Strong | No |

| No. | Outlook | Temperature | Humidity | Windy | d |
|---|---|---|---|---|---|
| 7 | Overcast | Middle | Middle | Strong | Yes |
| 8 | Sunny | Low | High | Weak | No |
| 9 | Sunny | Middle | Middle | Weak | Yes |
| 10 | Rain | Middle | Middle | Weak | Yes |
| 11 | Sunny | Middle | Middle | Strong | Yes |
| 12 | Overcast | Middle | High | Strong | Yes |
| 13 | Overcast | High | Middle | Weak | Yes |
| 14 | Rain | Middle | High | Strong | No |

Từ bảng quyết định không dư thừa thuộc tính 2.2, tìm tất cả các rút gọn $RED(C)$ của bảng 2.2 với $C = \{othw\}$ cũng chính là tất cả các rút gọn $RED(C)$ của bảng 2.1 với $C = \{ogthwn\}$

Theo định nghĩa 1.2.4, ta xây dựng phân hoạch lớp tương đương của tất cả các tổ hợp của bộ thuộc tính $C = \{othw\}$ như sau:

$U/\{o\} = \{\{1, 2, 8, 9, 11\}, \{3, 7, 12, 13\}, \{4, 5, 6, 10, 14\}\}$,

$U/\{t\} = \{\{1, 2, 3, 13\}, \{4, 7, 9, 10, 11, 12, 14\}, \{5, 6, 8\}\}$,

$U/\{h\} = \{\{1, 2, 3, 4, 8, 12, 14\}, \{5, 6, 7, 9, 10, 11, 13\}\}$,

$U/\{w\} = \{\{1, 3, 4, 5, 8, 9, 10, 13\}, \{2, 6, 7, 11, 12, 14\}\}$,

$U/\{d\} = \{\{1, 2, 6, 8, 14\}, \{3, 4, 5, 7, 9, 10, 11, 12, 13\}\}$,

$U/\{ot\} = \{\{1, 2\}, \{3, 13\}, \{4, 10, 14\}, \{5, 6\}, \{7, 12\}, \{8\}, \{9, 11\}\}$,

$U/\{oh\} = \{\{1, 2, 8\}, \{3, 12\}, \{4, 14\}, \{5, 6, 10\}, \{7, 13\}, \{9, 11\}\}$,

$U/\{ow\} = \{\{1, 8, 9\}, \{2, 11\}, \{3, 13\}, \{4, 5, 10\}, \{6, 14\}, \{7, 12\}\}$,

$U/\{th\} = \{\{1, 2, 3\}, \{4, 12, 14\}, \{5, 6\}, \{7, 9, 10, 11\}, \{8\}, \{13\}\}$,

$U/\{tw\} = \{\{1, 3, 13\}, \{2\}, \{4, 9, 10\}, \{5, 8\}, \{6\}, \{7, 11, 12, 14\}\}$,

$U/\{hw\} = \{\{1, 3, 4, 8\}, \{2, 12, 14\}, \{5, 9, 10, 13\}, \{6, 7, 11\}\}$,

$U/\{oth\} = \{\{1, 2\}, \{3\}, \{4, 14\}, \{5, 6\}, \{7\}, \{8\}, \{9, 11\}, \{10\}, \{12\}, \{13\}\}$,

$U/\{otw\} = \{\{1\},\{2\},\{3\},\{4,10\},\{5\},\{6\},\{7,12\},\{8\},\{9\},\{11\},\{13\},\{14\}\},$
$U/\{ohw\} = \{\{1,8\},\{2\},\{3\},\{4\},\{5,10\},\{6\},\{7\},\{9\},\{11\},\{12\},\{13\},\{14\}\},$
$U/\{thw\} = \{\{1,3\},\{2\},\{4\},\{5\},\{6\},\{7,11\},\{8\},\{9,10\},\{12,14\},\{13\}\}$

Dựa trên định nghĩa miền dương 1.2.5 của tập thuộc tính $C = \{othw\}$ ta tính được:

$POS_o(\{d\}) = \{3,7,12.13\},$
$POS_t(\{d\}) = \varnothing,$
$POS_h(\{d\}) = \varnothing,$
$POS_w(\{d\}) = \varnothing,$
$POS_{ot}(\{d\}) = \{1,2,3,7,8,9,11,12,13\},$
$POS_{oh}(\{d\}) = \{1,2,3,7,8,9,11,12,13\},$
$POS_{ow}(\{d\}) = \{3,4,5,6,7,12,13,14\},$
$POS_{th}(\{d\}) = \{7,8,9,10,11,13\},$
$POS_{tw}(\{d\}) = \{2,4,6,9,10\},$
$POS_{hw}(\{d\}) = \{5,9,10,13\},$
$POS_{oth}(\{d\}) = \{1,2,3,7,8,9,10,11,12,13\},$
$POS_{otw}(\{d\}) = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14\},$
$POS_{ohw}(\{d\}) = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14\},$
$POS_{thw}(\{d\}) = \{2,4,5,6,7,8,9,10,11,13\}$

Dễ dàng thấy được rằng $POS_{\{otw\}}(\{d\}) = POS_{\{ohw\}}(\{d\}) = U$. Theo định nghĩa 1.2.6, tập tất cả các rút gọn miền dương của tập thuộc tính $C = \{othw\}$ trên bảng quyết định nhất quán $DS = \{U, C \cup \{d\}, V, f\}$ (bảng 2.2) là $RED(C) = \{otw, ohw\}$ cũng chính là bằng tập tất cả các rút gọn miền dương của tập thuộc tính $C = \{ogthwn\}$ trên bảng quyết định nhất quán $DS$ (bảng 2.1). Như vậy ở đây thấy rõ ràng hai thuộc tính dư thừa "Grass, NumberHoles" không gây bất kỳ ảnh hưởng nào đến việc tìm tập tất cả các tập rút gọn $RED(C)$. Từ đây chúng ta chỉ cần sử dụng bảng quyết định nhất quán 2.2 thay cho bảng quyết định nhất quán 2.1.

Tiếp theo, đối với một bảng quyết định nhất quán bất kỳ đều có thể giảm kích thước theo chiều dọc bảng theo như thuật toán *AnObjectReduct($DS$)* để tìm rút gọn đối tượng. Mục tiêu thứ nhất tìm rút gọn đối tượng từ bảng quyết định nhất quán là để giảm kích thước lưu trữ. Mục tiêu thứ hai của tìm rút gọn đối tượng từ bảng quyết định nhất quán là phải giữ được tập tất cả các tập rút gọn miền dương $RED(C)$ theo định nghĩa 1.2.6. Hai mục tiêu này cũng là hai mục tiêu của việc loại bỏ tất cả các thuộc tính dư thừa như thuật toán *RemoveRedundantAttribute($DS$)* chỉ khác nhau ở tính chất là rút gọn đối tượng là thu gọn các hàng của bảng quyết định nhất quán, còn loại bỏ các thuộc tính dư thừa là thu gọn các cột của bảng quyết định nhất quán.

Từ định nghĩa 1.1.6, đối với mỗi cặp hàng $(i, j)$ của bảng quyết định nhất quán 2.2, trước hết phải xây dựng các $E_{ij}$. $E_{1,2} = \{othd\}$, $E_{1,3} = \{thw\}$, tương tự như vậy với các cặp $(1, 4)$, ..., $(1, 14)$, $(2, 3)$, $(2, 4)$, ..., $(13, 14)$ đạt được tập $E_r$ chứa tất cả các tập $A_i$ như sau:

$A_1 = \{othd\} = E_{1,2} = E_{9,11}$,

$A_2 = \{thw\} = E_{1,3} = E_{12,14}$,

$A_3 = \{hw\} = E_{1,4} = E_{2,12} = E_{3,8} = E_{4,8} = E_{6,7} = E_{6,11}$,

$A_4 = \{w\} = E_{1,5} = E_{1,10} = E_{2,7} = E_{6,12} = E_{8,10} = E_{8,13}$,

$A_5 = \{d\} = E_{1,6} = E_{3,11} = E_{5,12}$,

$A_6 = \{ohwd\} = E_{1,8} = E_{5,10}$,

$A_7 = \{ow\} = E_{1,9} = E_{2,11} = E_{8,9}$,

$A_8 = \{o\} = E_{1,11} = E_{2,9} = E_{4,6} = E_{5,14} = E_{8,11}$,

$A_9 = \{h\} = E_{1,12} = E_{2,4} = E_{3,14} = E_{6,9} = E_{6,13} = E_{8,12}$,

$A_{10} = \{tw\} = E_{1,13} = E_{5,8} = E_{7,14} = E_{11,14}$,

$A_{11} = \{hd\} = E_{1,14} = E_{5,7} = E_{5,11} = E_{8,14} = E_{11,13}$,

$A_{12} = \{th\} = E_{2,3}$,

$A_{13} = \{wd\} = E_{2,6} = E_{3,5} = E_{3,9} = E_{3,10} = E_{4,13}$,

$A_{14} = \{ohd\} = E_{2,8} = E_{3,12} = E_{7,13}$,

$A_{15} = \{t\} = E_{2,13} = E_{9,14}$,

$A_{16} = \{hwd\} = E_{2,14} = E_{3,4} = E_{5,9} = E_{5,13} = E_{9,13} = E_{10,13}$,

$A_{17} = \{od\} = E_{3,7} = E_{12,13}$,

$A_{18} = \{otwd\} = E_{3,13} = E_{4,10} = E_{7,12}$,

$A_{19} = \{owd\} = E_{4,5} = E_{6,14}$,

$A_{20} = \{td\} = E_{4,7} = E_{4,11} = E_{6,8} = E_{9,12} = E_{10,12}$,

$A_{21} = \{twd\} = E_{4,9} = E_{11,12}$,

$A_{22} = \{thd\} = E_{4,12} = E_{7,9} = E_{7,10} = E_{10,11}$,

$A_{23} = \{oth\} = E_{4,14} = E_{5,6}$,

$A_{24} = \{oh\} = E_{6,10}$,

$A_{25} = \{thwd\} = E_{7,11} = E_{9,10}$,

$A_{26} = \{ot\} = E_{10,14}$

Cuối cùng đạt được $E_r = \{A_1, ..., A_{26}\} = E_r^U$. Bước 1 thuật toán *AnObjectReduct(*$DS$*)* có $E_r$. Bước 2 xây dựng tập $M_d^U$ theo định nghĩa 1.2.3 hệ bằng nhau của tập thuộc tính $d$ đạt được:

$M_d = M_d^U = \{thw, oth, ow\} = \{B_1, B_2, B_3\}$

Tiếp theo, bước 3 và bước 4 với $T(0) = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$, $E_r$, $M_d^U$ tính được:

$M_d^{T(0)-\{1\}} = \{thw, oth, ow\} = M_d^U \Rightarrow T(1) = T(0) - \{1\}$,

$M_d^{T(1)-\{2\}} = \{thw, oth, ow\} = M_d^U \Rightarrow T(2) = T(1) - \{2\}$,

$M_d^{T(2)-\{3\}} = \{thw, oth, ow\} = M_d^U \Rightarrow T(3) = T(2) - \{3\}$,

$M_d^{T(3)-\{4\}} = \{thw, oth, ow\} = M_d^U \Rightarrow T(4) = T(3) - \{4\}$,

$M_d^{T(4)-\{5\}} = \{thw, ow, oh, ot\} \neq M_d^U \Rightarrow T(5) = T(4)$,

$M_d^{T(5)-\{6\}} = \{thw, ow, ot\} \neq M_d^U \Rightarrow T(6) = T(5)$,

$M_d^{T(6)-\{7\}} = \{thw, oth, ow\} = M_d^U \Rightarrow T(7) = T(6) - \{7\}$,

$M_d^{T(7)-\{8\}} = \{thw, oth\} \neq M_d^U \Rightarrow T(8) = T(7)$,

$M_d^{T(8)-\{9\}} = \{thw, oth\} \neq M_d^U \Rightarrow T(9) = T(8)$,

$M_d^{T(9)-\{10\}} = \{thw, oth, ow\} = M_d^U \Rightarrow T(10) = T(9) - \{10\},$
$M_d^{T(10)-\{11\}} = \{thw, oth, ow\} = M_d^U \Rightarrow T(11) = T(10) - \{11\},$
$M_d^{T(11)-\{12\}} = \{oth, ow, tw\} \neq M_d^U \Rightarrow T(12) = T(11),$
$M_d^{T(12)-\{13\}} = \{thw, oth, ow\} = M_d^U \Rightarrow T(13) = T(12) - \{13\},$
$M_d^{T(13)-\{14\}} = \{oth, ow, tw\} \neq M_d^U \Rightarrow T(14) = T(13)$

Đặt $U' = T(14) = \{5, 6, 8, 9, 12, 14\}$ thì $DS' = (U', C \cup \{d\}, V, f)$ là rút gọn đối tượng của bảng quyết định nhất quán 2.2 là bảng quyết định nhất quán 2.3.

Bảng 2.3: Một rút gọn đối tượng của bảng quyết định nhất quán 2.2

| No. | Outlook | Temperature | Humidity | Windy | d |
|---|---|---|---|---|---|
| 5 | Rain | Low | Middle | Weak | Yes |
| 6 | Rain | Low | Middle | Strong | No |
| 8 | Sunny | Low | High | Weak | No |
| 9 | Sunny | Middle | Middle | Weak | Yes |
| 12 | Overcast | Middle | High | Strong | Yes |
| 14 | Rain | Middle | High | Strong | No |

Từ rút gọn đối tượng của bảng quyết định nhất quán 2.2 là bảng 2.3, kiểm tra tập tất cả các tập rút gọn $RED(C)$ miền dương của bảng 2.2 có đúng là bằng với $RED(C)$ của bảng 2.3. Giống như các bước trước, tìm phân hoạch các lớp tương đương trên bảng 2.3 đạt được:

$U'/\{o\} = \{\{5, 6, 14\}, \{8, 9\}, \{12\}\},$
$U'/\{t\} = \{\{5, 6, 8\}, \{9, 12, 14\}\},$
$U'/\{h\} = \{\{5, 6, 9\}, \{8, 12, 14\}\},$
$U'/\{w\} = \{\{5, 8, 9\}, \{6, 12, 14\}\},$
$U'/\{d\} = \{\{5, 9, 12\}, \{6, 8, 14\}\},$
$U'/\{ot\} = \{\{5, 6\}, \{8\}, \{9\}, \{12\}, \{14\}\},$

$U'/\{oh\} = \{\{5,6\},\{8\},\{9\},\{12\},\{14\}\}$,

$U'/\{ow\} = \{\{5\},\{6,14\},\{8,9\},\{12\}\}$,

$U'/\{th\} = \{\{5,6\},\{8\},\{9\},\{12,14\}\}$,

$U'/\{tw\} = \{\{5,8\},\{6\},\{9\},\{12,14\}\}$,

$U'/\{hw\} = \{\{5,9\},\{6\},\{8\},\{12\},\{14\}\}$,

$U'/\{oth\} = \{\{5,6\},\{8\},\{9\},\{12\},\{14\}\}$,

$U'/\{otw\} = \{\{5\},\{6\},\{8\},\{9\},\{12\},\{14\}\}$,

$U'/\{ohw\} = \{\{5\},\{6\},\{8\},\{9\},\{12\},\{14\}\}$,

$U'/\{thw\} = \{\{5\},\{6\},\{8\},\{9\},\{12,14\}\}$,

$POS'_o(\{d\}) = \{12\}$,

$POS'_t(\{d\}) = \varnothing$,

$POS'_h(\{d\}) = \varnothing$,

$POS'_w(\{d\}) = \varnothing$,

$POS'_{ot}(\{d\}) = \{8,9,12,14\}$,

$POS'_{oh}(\{d\}) = \{8,9,12,14\}$,

$POS'_{ow}(\{d\}) = \{5,6,12,14\}$,

$POS'_{th}(\{d\}) = \{8,9\}$,

$POS'_{tw}(\{d\}) = \{6,9\}$,

$POS'_{hw}(\{d\}) = \{5,6,8,9,12,14\}$,

$POS'_{oth}(\{d\}) = \{8,9,12,14\}$,

$POS'_{otw}(\{d\}) = \{5,6,8,9,12,14\}$,

$POS'_{ohw}(\{d\}) = \{5,6,8,9,12,14\}$,

$POS'_{thw}(\{d\}) = \{5,6,8,9\}$

Có thể thấy rằng $POS'_{\{hw\}}(\{d\}) = POS'_{\{otw\}}(\{d\}) = POS'_{\{ohw\}}(\{d\}) = U'$. Đặt $P = \{POS'_B(\{d\})\} = \{hw, otw, ohw\}$. Vì $U'$ là một rút gọn đối tượng của $U$, theo định nghĩa 1.2.6 thì tập tất cả các rút gọn của $C$ phải là hệ Sperner nên $RED_{U'}(C) = \{B \in P, \nexists A \in P, A \subset B\}$ mà rõ ràng trong $P$ tồn tại $hw \subset ohw$ không thỏa mãn hệ

Sperner nên phải xóa bỏ $hw$ khỏi $P$ thì $P$ trở thành một hệ Sperner. Đặt $RED_{U'}(C) = P - \{hw\} = \{otw, ohw\}$. Rõ ràng là $RED_{U'}(C) = RED_U(C)$ với $RED_{U'}(C)$ là tập tất cả các rút gọn miền dương của $DS' = (\{5, 6, 8, 9, 12, 14\}, \{othw\} \cup \{d\}, V, f)$ và $RED_U(C)$ là tập tất cả các rút gọn miền dương của $DS = (U, \{othw\} \cup \{d\}, V, f)$.

Khác với sử dụng rút gọn miền dương theo định nghĩa 1.2.6 để tìm tập tất cả các tập rút gọn $RED(C)$ của $C = \{othw\}$ có độ phức tạp tính toán thời gian NP-complete, thuật toán *AnAttributeReduct($DS$)* sẽ tìm một rút gọn theo miền dương của $C$ từ bảng quyết định nhất quán 2.2 với độ phức tạp tính toán thời gian đa thức:
$temp = H(0) - \{o\} = \{thw\} = B_1 \in M_d \Rightarrow H(1) = H(0)$,
$temp = H(1) - \{t\} = \{ohw\} \notin \{B \in M_d\} \Rightarrow H(2) = temp$,
$temp = H(2) - \{h\} = \{ow\} = B_3 \in M_d \Rightarrow H(3) = H(2)$,
$temp = H(3) - \{w\} = \{oh\} \subseteq B_2 \in M_d \Rightarrow H(4) = H(3)$

Đặt $H = H(4) = \{ohw\}$ và thuật toán *AnAttributeReduct($DS$)* dừng, khi đó $H \in RED(C)$. Đạt được một bảng rút gọn thuộc tính theo miền dương là bảng 2.4 từ bảng quyết định nhất quán 2.2. Tuy nhiên, bảng quyết định rút gọn thuộc tính này không còn là nhất quán vì có hàng $1, 8$ trùng nhau và hàng $5, 10$ trùng nhau. Mặc dù vậy, bảng rút gọn thuộc tính 2.4 vẫn sinh ra các luật quyết định đúng theo các luật quyết định được sinh ra từ bảng 2.2.

Bảng 2.4: Một rút gọn thuộc tính miền dương của bảng 2.2

| No. | Outlook | Humidity | Windy | d |
|---|---|---|---|---|
| $1, 8$ | Sunny | High | Weak | No |
| $2$ | Sunny | High | Strong | No |
| $3$ | Overcast | High | Weak | Yes |
| $4$ | Rain | High | Weak | Yes |
| $5, 10$ | Rain | Middle | Weak | Yes |

| No. | Outlook | Humidity | Windy | d |
|---|---|---|---|---|
| 6 | Rain | Middle | Strong | No |
| 7 | Overcast | Middle | Strong | Yes |
| 9 | Sunny | Middle | Weak | Yes |
| 11 | Sunny | Middle | Strong | Yes |
| 12 | Overcast | High | Strong | Yes |
| 13 | Overcast | Middle | Weak | Yes |
| 14 | Rain | High | Strong | No |

Kết hợp hai thuật toán *AnAttributeReduct($DS$)* và *AnObjectReduct($DS$)* sẽ thu được bảng quyết định rút gọn thuộc tính miền dương và thu gọn đối tượng mà vẫn bảo toàn các luật quyết định được sinh ra là đúng và ngắn gọn có nghĩa là sinh ra được cây quyết định nhỏ (có thể là nhỏ nhất cũng có thể không nhỏ nhất nhưng vẫn là nhỏ). Từ bảng 2.2 áp dụng hai thuật toán này thu được bảng quyết định nhất quán với tập vũ trụ $U_1 = \{5, 6, 8, 9, 12, 14\}$ trên tập thuộc tính $\{ohw\}$ là thu gọn theo cả chiều ngang và chiều dọc đảm bảo sinh ra đủ một bộ luật quyết định như kết quả là bảng 2.5.

Bảng 2.5: Kết hợp rút gọn đối tượng và thuộc tính của bảng 2.2

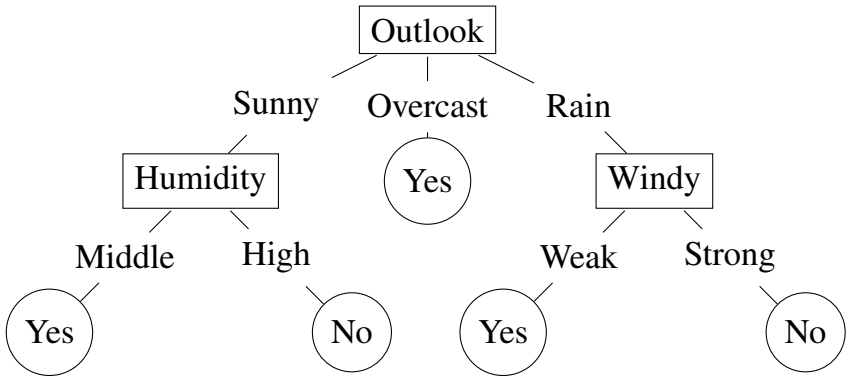
| No. | Outlook | Humidity | Windy | d |
|---|---|---|---|---|
| 1 | Rain | Middle | Weak | Yes |
| 2 | Rain | Middle | Strong | No |
| 3 | Sunny | High | Weak | No |
| 4 | Sunny | Middle | Weak | Yes |
| 5 | Overcast | High | Strong | Yes |
| 6 | Rain | High | Strong | No |

Một cây quyết định được sinh ra từ bảng quyết định nhất quán tối thiểu 2.5 là sự kết hợp hai thuật toán *AnObjectReduct($DS$)* và *AnAttributeReduct($DS$)* từ bảng quyết

định nhất quán 2.2. Ma trận phân biệt tìm được theo định nghĩa 1.2.7 trên bảng 2.3 vì $|U'| = 6 < 14 = |U|$.

$$\begin{bmatrix} \varnothing & & & & & \\ w & \varnothing & & & & \\ oh & \varnothing & \varnothing & & & \\ \varnothing & otw & th & \varnothing & & \\ \varnothing & oth & otw & \varnothing & \varnothing & \\ thw & \varnothing & \varnothing & ohw & o & \varnothing \end{bmatrix}$$

Tập CORE được tính theo định nghĩa 1.2.8 $CORE(C) = \{w, o\}$ sử dụng thuật toán *DecisionTree(DS)* sinh ra cây quyết định như hình 2.1. Dễ dàng kiểm chứng được cây quyết định này trùng với trùng với cây quyết định được sinh ra theo thuật toán ID3 từ bảng quyết định nhất quán 2.1.




Hình 2.1: Cây quyết định được sinh ra từ thuật toán *DecisionTree(DS)*

## 2.7 Đánh giá thực nghiệm

Trong phần này, nghiên cứu sinh chứng tỏ bằng thực nghiệm kết quả đánh giá các thuật toán nghiên cứu sinh đề xuất trong chương khai phá dữ liệu dạng bảng. Các kết quả thực nghiệm được thực hiện nhanh với ngôn ngữ lập trình Nodejs, Javascript với một số tập dữ liệu dạng txt từ kho dữ liệu UCI.

Bảng 2.6: Bảng thực hiện một rút gọn thuộc tính

| Tập dữ liệu | Thuộc tính gốc | Thuộc tính rút gọn | Thời gian(s) |
|---|---|---|---|
| Examples | 4 | 3 | 0.006 |
| Breast cancer | 9 | 8 | 0.161 |
| Balance | 4 | 3 | 0.248 |
| Car Evaluation | 6 | 5 | 0.673 |

Bảng 2.7: Bảng thực hiện rút gọn đối tượng

| Tập dữ liệu | Đối tượng gốc | Đối tượng rút gọn | Thời gian(s) |
|---|---|---|---|
| Examples | 14 | 6 | 0.005 |
| Breast cancer | 286 | 2 | 0.158 |
| Balance | 625 | 6 | 0.171 |
| Car Evaluation | 1728 | 9 | 0.771 |

Thực nghiệm chỉ ra rằng tốc độ tính toán của thuật toán IRDT là nhanh vượt trội so với thuật toán ID3. Có thể dễ dàng nhận thấy rằng vấn đề đếm số lượng phần tử trong các tập quan hệ bất khả phân biệt là các phép tính toán số nguyên nên rõ ràng nhanh hơn tính hàm độ đo thông tin (Entropy và tính Information Gain) vốn là các công thức tính toán số thực.

Bảng 2.8: Bảng so sánh tốc độ thực hiện IDRT và ID3 (millisecond)

| Datasets (Atts/Objs) | ID3 (ms) | IRDT (ms) |
|---|---|---|
| Examples (4/14) | 3 | 1 |
| Breast cancer (9/286) | 53 | 13 |
| Car Evaluation (6/1728) | 64 | 30 |

Phân tích rõ hơn về tốc độ thực hiện của thuật toán IRDT nhanh hơn thuật toán

ID3 một số điểm như sau:

Lấy ví dụ với bảng quyết định nhất quán 2.2, IRDT sử dụng các phân hoạch tương đương của các thuộc tính đếm số phần tử mà phân hoạch của một thuộc tính bất kỳ $A$ nằm trong phân hoạch của thuộc tính quyết đinh $d$.

Phân hoạch tương đương của thuộc tính quyết định $d$: $\{1, 2, 6, 8, 14\}$ có 5 phần tử No và $\{3, 4, 5, 7, 9, 10, 11, 12, 13\}$ có 9 phần tử Yes.

Phân hoạch tương đương của thuộc tính Outlook: $\{1, 2, 8, 9, 11\}$ có 5 phần tử Sunny, $\{3, 7, 12, 13\}$ có 4 phần tử Overcast và $\{4, 5, 6, 10, 14\}$ có 5 phần tử Rain

Phân hoạch tương đương của thuộc tính Temperature: $\{1, 2, 3, 13\}$ có 4 phần tử High, $\{4, 7, 9, 10, 11, 12, 14\}$ có 7 phần tử Middle và $\{5, 6, 8\}$ có 3 phần tử Low.

Phân hoạch tương đương của thuộc tính Hummidity: $\{1, 2, 3, 4, 8, 12, 14\}$ có 7 phần tử High và $\{5, 6, 7, 9, 10, 11, 13\}$ có 7 phần tử Middle.

Phân hoạch tương đương của thuộc tính Windy: $\{1, 3, 4, 5, 8, 9, 10, 13\}$ có 8 phần tử Weak và $\{2, 6, 7, 11, 12, 14\}$ có 6 phần tử Strong.

So sánh giữa hai thuật toán IRDT và ID3: IRDT chỉ đếm phần số phần tử max trong phân hoạch của thuộc tính để chọn thuộc tính tốt nhất trong khi ID3 sử dụng phép tính logarit để xác định thuộc tính tốt nhất.

- IRDT chọn thuộc tính Outlook làm thuộc tính gốc của cây quyết định vì trong phân hoạch của Outlook có $\{3, 7, 12, 13\}$ là tập con của $\{3, 4, 5, 7, 9, 10, 11, 12, 13\}$ trong phân hoạch của $d$. Các thuộc tính khác như Temperature, Humidity, Windy không có tập nào trong phân hoạch là tập con của một tập trong phân hoạch của $d$.

- ID3 sẽ tính hàm độ đo thông tin (Information Gain) cho các thuộc tính bằng công thức logarit cho các thuộc tính Outlook, Temperature, Humidity, Windy. Ví dụ về tính hàm độ đo thông tin (Information Gain) cho thuộc tính Outlook như sau:

$$H(d) = -\frac{5}{14}log\frac{5}{14} - \frac{9}{14}log\frac{9}{14}$$

$$InfoGain(Outlook) = H(d) - \frac{5}{14}O_S - \frac{4}{14}O_O - \frac{5}{15}O_R$$

$$O_S = -\frac{3}{5}log\frac{3}{5} - \frac{2}{5}log\frac{2}{5}$$

$$O_O = -\frac{4}{4}log\frac{4}{4} - \frac{0}{4}log\frac{0}{4}$$

$$O_R = -\frac{3}{5}log\frac{3}{5} - \frac{2}{5}log\frac{2}{5}$$

$InfoGain(Outlook) = 0.247$

$InfoGain(Temperature) = 0.029$

$InfoGain(Hummidity) = 0.152$

$InfoGain(Windy) = 0.048$

Các bước tiếp theo sẽ giống như bước đầu tiên sử dụng phép đếm đối với thuật toán IRDT và sử dụng phép tính số thực với ID3 cho tập đối tượng đã bị rút nhỏ lại sau bước tính toán trước đó.

Từ kết quả trên có thể thấy rõ rằng nếu thực hiện hai thuật toán IRDT và ID3 trên máy tính thì phép đếm số nguyên luôn thực hiện với tốc độ nhanh hơn so với tính toán số thực của thuật toán ID3.

Bộ dữ liệu trong thực nghiệm được lấy từ kho dữ liệu UCI như Breast Cancer (các bệnh nhân ung thư vú có 286 đối tượng và 9 thuộc tính được phân thành 2 lớp với một lớp có 201 đối tượng và lớp còn lại có 85 đối tượng phân loại tái phát hay không tái phát), Balance (tập dữ liệu sinh ra mô hình kết quả thí nghiệm tâm lý có 625 đối tượng và 4 thuộc tính gồm: trọng lượng bên trái, khoảng cách bên trái, trọng lượng bên phải, khoảng cách bên phải phân loại trái, phải hay cân bằng), Car Evaluation (mô hình đánh giá xe ô tô theo cấu trúc khái niệm có 1728 đối tượng và 6 thuộc tính hỗ trợ ra quyết định chấp nhận, không chấp nhận, tốt, rất tốt) và Examples (tập dữ liệu mẫu về ra quyết định đi chơi golf hay không với 14 đối tượng và 6 thuộc tính). Các tập dữ liệu được lấy ngẫu nhiên phù hợp với khai phá dữ liệu trên bảng quyết định đầy đủ và nhất quán.

## 2.8 Kết luận chương

Rút gọn thuộc tính là một mục tiêu quan trọng hàng đầu trong khai phá dữ liệu bảng quyết định. Trong bối cảnh dữ liệu ngày càng tăng lên về số lượng đối tượng, vấn đề rút gọn thuộc tính bị ảnh hưởng kéo theo tốc độ xử lý không đáp ứng được nhu cầu thực tiễn. Vấn đề tìm tất cả các rút gọn thuộc tính có độ phức tạp thời gian hàm mũ là không khả thi, các công trình công bố sử dụng nhiều phương pháp để tìm một rút gọn thuộc tính theo heuristic. Dù sử dụng phương pháp heuristic để tìm được một rút gọn thuộc tính thì bản chất độ phức tạp tính toán vẫn phải duyệt toàn bộ đối tượng trong bảng quyết định. Trong chương này, nghiên cứu sinh đề xuất phương pháp tìm một rút gọn đối tượng trong thời gian đa thức mà kết quả là thu được bảng quyết định với số đối tượng ít hơn nhiều lần so với bảng quyết định nhất quán gốc và vẫn đảm bảo quá trình tìm các rút gọn thuộc tính không bị ảnh hưởng. Thêm vào đó, nghiên cứu sinh cũng đề xuất một phương pháp tìm một rút gọn thuộc tính không heuristic và một phương pháp cải tiến sinh cây quyết định từ bảng quyết định nhất quán với tốc độ thực hiện nhanh hơn thuật toán sinh cây quyết định ID3. Các đề xuất của nghiên

cứu sinh được thực hiện trong thời gian đa thức, được chứng minh tính đúng đắn và đầy đủ theo lý thuyết cùng thực nghiệm.

# CHƯƠNG 3

# KHAI PHÁ DỮ LIỆU ĐỒ THỊ

Trong chương này, nghiên cứu sinh trình bày về thuật toán đề xuất là kết quả mới của luận án về khai phá đồ thị con thường xuyên đóng trong đó chứng minh vấn đề đẳng cấu đồ thị con được giải quyết trong thời gian đa thức. Thêm nữa, nghiên cứu sinh cũng trình bày kết quả mới của luận án về độ đo tương tự trên dàn giao khái niệm sử dụng trong vấn đề phân loại đa nhãn cho đồ thị.

Nội dung chương này dựa trên các công trình số [1], [3], [5], [7] trong danh mục công trình công bố của nghiên cứu sinh.

## 3.1 Đặt vấn đề

Dữ liệu đồ thị là cấu trúc dữ liệu phức tạp hơn nhiều lần dữ liệu dạng bảng. Do đó, các công trình khai phá dữ liệu trên đồ thị gặp nhiều khó khăn với độ phức tạp thời gian không đa thức. Nhiều vấn đề có thể được giải quyết trong bảng với thuật toán độ phức tạp thời gian đa thức nhưng khi chuyển sang dạng dữ liệu đồ thị thì chỉ có thể giải quyết bằng các thuật toán độ phức tạp thời gian không đa thức. Bằng một số các phép biến đổi, các thuật toán khai phá dữ liệu đồ thị có thể tối ưu để giảm thời gian tính toán trong quá trình khai phá dữ liệu. Các phép biến đổi này đôi khi sẽ áp dụng một số điều kiện ràng buộc đặc trưng cho tập dữ liệu cụ thể và mục tiêu khai phá dữ liệu để đạt được tối ưu về mặt giảm thời gian tính nhưng có thể sẽ phải trả giá bằng tăng không gian tính toán. Chương này đặt trọng tâm vào tối ưu thời gian tính toán cho một số bài toán khai phá dữ liệu đồ thị cụ thể như khai phá đồ thị con thường xuyên và phân loại đa nhãn cho đồ thị.

Thứ nhất, khai phá mẫu thường xuyên là đi tìm tất cả các mẫu là một biểu diễn

cấu trúc con của một biểu diễn cấu trúc cha (chẳng hạn như biểu diễn cấu trúc tập hợp trong khai phá tập mục thường xuyên [3], biểu diễn cấu trúc cây trong khai phá cây con thường xuyên, biểu diễn đồ thị trong khai phá đồ thị con thường xuyên [44], [46], [52], [88], [89] thỏa mãn tần suất xuất hiện của mẫu con này lớn hơn một ngưỡng mà người dùng tự định nghĩa [37]. Tìm các cấu trúc con thường xuyên đóng vai trò quan trọng trong khai phá dữ liệu kết hợp, tương quan và nhiều quan hệ khác của dữ liệu. Hơn nữa, các mẫu thường xuyên trợ giúp trong việc đánh chỉ mục dữ liệu, phân lớp, phân cụm, và các nhiệm vụ khai phá dữ liệu khác. Như vậy, khai phá mẫu thường xuyên trở thành nhiệm vụ khai phá dữ liệu quan trọng và trọng tâm trong lĩnh vực khai phá dữ liệu. Khai phá mẫu thường xuyên dựa trên một tính chất "Downward Closure Property" hay còn được gọi là tính chất phản đơn điệu. Tính chất này thể hiện rằng nếu mẫu cha thỏa mãn tính chất thường xuyên thì tất cả các mẫu con của nó cũng thỏa mãn tính chất thường xuyên và nếu mẫu con là không thỏa mãn tính chất thường xuyên thì mọi mẫu cha của nó cũng không thỏa mãn tính chất thường xuyên. Khai phá mẫu thường xuyên hiện nay có hai hướng tiếp cận là tuân theo nguyên lý Apriori [46] hoặc theo nguyên lý FP-Growth [88]. Dù có tuân theo nguyên lý nào thì khai phá mẫu thường xuyên cũng có độ phức tạp tính toán thời gian thuộc lớp không đa thức NP.

Sự giống nhau giữa khai phá mẫu tập mục thường xuyên và khai phá mẫu đồ thị con thường xuyên ở vấn đề sinh và kiểm tra một ứng viên là tập mục con hay đồ thị con có thỏa mãn tính chất thường xuyên hay không. Sự khác nhau chính là ở khai phá mẫu tập mục thường xuyên thì mỗi tập mục ứng viên được sinh ra chỉ có duy nhất một mình nó vấn đề kiểm tra nó có nằm trong một tập mục giao tác hay không chỉ thực hiện trong thời gian đa thức, nhưng trong khai phá mẫu đồ thị con thường xuyên thì vấn đề sinh một đồ thị con ứng viên lại không phải là duy nhất và vấn đề kiểm tra một đồ thị con ứng viên có nằm trong một đồ thị giao tác hay không chính là vấn đề đẳng cấu đồ thị con và vấn đề này được các nhà khoa học xác định là phải tính toán trong thời gian có độ phức tạp thuộc lớp không đa thức đầy đủ NP-complete [31]. Để

giải quyết tối ưu thời gian tính vấn đề đẳng cấu đồ thị con, nghiên cứu sinh thực hiện một số điều kiện ràng buộc về thứ tự nhãn của đỉnh và cạnh cùng với biểu diễn chuẩn hóa cùng kỹ thuật máy ngẫu nhiên để đạt thời gian tính toán tối ưu nhất.

Thứ hai, học máy là lĩnh vực rất quan trọng trong khai phá dữ liệu. Các dữ liệu ngày càng đa dạng về mặt cấu trúc, các phương pháp khai phá dữ liệu trên bảng gặp nhiều khó khăn như dữ liệu cấu trúc tế bào, cấu trúc mạng nơron, cấu trúc protein, cấu trúc dữ liệu mạng máy tính, mạng xã hội. Để giải quyết các vấn đề này các nhà khoa học đã áp dụng biểu diễn dữ liệu cấu trúc đồ thị, cây, dàn giao và áp dụng các phương pháp khai phá dữ liệu hiện có với các loại biểu diễn dữ liệu khác lên các biểu diễn dữ liệu đồ thị. Học máy áp dụng trên đồ thị là một hướng đi đúng đắn cho xu thế dữ liệu đa dạng và phức tạp ngày nay.

Do tính chất đa dạng của dữ liệu và đa dạng các mục tiêu, các phương pháp phân lớp dữ liệu trong học máy dần chuyển từ phân loại đa lớp đơn nhãn sang phân loại đa lớp đa nhãn. Nhiều công trình công bố về phân loại đa nhãn trên biểu diễn dữ liệu dạng bảng, trong đó nhiều mô hình tính toán phân loại đa nhãn hiệu quả dựa trên lý thuyết Dempster Shafer. Mô hình phân loại đa nhãn dựa trên lý thuyết Dempster Shafer tăng độ chính xác phân loại, giảm thời gian thực hiện và ngày càng nhiều công trình công bố. Tuy nhiên để áp dụng phân loại đa nhãn cho đồ thị là khá khó khăn vì bản chất biểu diễn dữ liệu đồ thị khó có thể chuyển đổi về biểu diễn véctơ để áp dụng các thuật toán phân loại đa nhãn. Các công nghệ thu thập dữ liệu ngày càng được cải tiến, nhiều lĩnh vực ứng dụng phải đối mới với dữ liệu đa dạng và đa cấu trúc như cấu trúc hóa học, cấu trúc luồng chương trình, tài liệu XML, web. Khác với dữ liệu truyền thống trong không gian đặc trưng, các dữ liệu này không thể biểu diễn dưới dạng véctơ mà chỉ có thể biểu diễn dưới dạng đồ thị dẫn đến một thách thức cơ bản của khai phá dữ liệu đồ thị là thiếu biểu diễn véctơ dẫn đến khó xác định được độ đo tương tự của đồ thị để áp dụng các mô hình phân loại đa nhãn dựa trên lý thuyết Dempster Shafer. Nghiên cứu sinh đề xuất một độ đo tương tự cho các dồ thị dựa trên

dàn giao khái niệm từ tập đồ thị con thường xuyên đóng và áp dụng độ đo tương tự này trên các mô hình phân loại đa nhãn đồ thị dựa trên lý thuyết Dempster Shafer. Nghiên cứu sinh cũng chứng minh lý thuyết độ đo tương tự trên dàn giao khái niệm là đúng đắn và đầy đủ.

## 3.2 Khai phá đồ thị con thường xuyên đóng

Khai phá đồ thị là một vùng chính được quan tâm trong lĩnh vực khai phá dữ liệu những năm gần đây. Một lớp con quan trọng của khai phá dữ liệu đồ thị là khai phá đồ thị con thường xuyên. Vấn đề trọng tâm của khai phá đồ thị con thường xuyên là khái niệm đẳng cấu đồ thị con. Nghiên cứu đẳng cấu đồ thị con trước đây chủ yếu là vấn đề về độ phức tạp tính toán. Thông thường, vấn đề đẳng cấu đồ thị con là vấn đề NP-complete [31]. Các công trình nghiên cứu trước đây về khai phá đồ thị con thường xuyên chưa giải quyết được vấn đề NP-complete trong đẳng cấu đồ thị con. Do đó, trong luận án này nghiên cứu sinh đề xuất thuật toán mới giải quyết vấn đề đẳng cấu đồ thị con trong thời gian đa thức trong khai phá dữ liệu đồ thị. Hơn nữa thuật toán mới cũng được chứng minh về mặt lý thuyết tính hiệu quả hơn các công trình trước đó trong khai phá đồ thị con thường xuyên đóng.

Thông thường, khai phá đồ thị con thường xuyên đặt mục tiêu vào xác định tất cả các mẫu đồ thị con mà có tần suất xuất hiện bên trong một tập dữ liệu đồ thị trên một ngưỡng người dùng định nghĩa. Những mẫu đồ thị con này được gọi là đồ thị con thường xuyên. Theo lý thuyết, khai phá đồ thị con thường xuyên có thể được công thức hóa như tìm kiếm trong không gian tìm kiếm, được mô hình như một dàn giao, gồm tất cả các mẫu đồ thị con có khả năng. Bởi vì số lượng các đồ thị con thường xuyên có thể tăng theo cấp số mũ với kích thước của đồ thị, duyệt đầy đủ không gian tìm kiếm là không thể tính toán được bởi nguyên do "bùng nổ tổ hợp". Một ngưỡng độ hỗ trợ người dùng tự định nghĩa thường được dùng để tỉa không gian tổ hợp để phân tách

các đồ thị con không thường xuyên và các đồ thị con thường xuyên. Khai phá đồ thị con thường xuyên đã nhận được nhiều sự quan tâm đáng kể [44],[46], [52],[88], [89]. Các kỹ thuật khai phá đồ thị con thường xuyên được phân vào hai hướng tiếp cận: (i) tiếp cận dựa trên Apriori (cũng được gọi là tiếp cận dựa trên chiến lược tìm kiếm theo bề rộng) [46] và (ii) tiếp cận dựa trên phát triển mẫu (cũng được gọi là tiếp cận dựa trên chiến lược tìm kiếm theo độ sâu) [88]. Cả hai hướng tiếp cận đều có những ưu điểm và khuyết điểm tuy nhiên chúng đều bao gồm sinh tập ứng viên và kiểm tra đẳng cấu đồ thị con để quyết định đồ thị con có phải là thường xuyên hay không. Trong lý thuyết khoa học máy tính, vấn đề đẳng cấu đồ thị con là tác vụ tính toán trong đó cho hai đồ thị con G và H, cần phải xác định G có chứa một đồ thị con mà đẳng cấu với H hay không. Một số trường hợp của đẳng cấu đồ thị có thể giải quyết trong thời gian đa thức [29], [40] cho các đồ thị phẳng.

Một số lượng lớn các công trình đã được công bố về đẳng cấu đồ thị con trong vấn đề khai phá đồ thị con thường xuyên [29], [40], [44], [58], [84]. Có một số công trình nghiên cứu chỉ ra đẳng cấu đồ thị con thường xuyên trong thời gian đa thức cho đồ thị phẳng [29], [40] nhưng hầu hết các đồ thị trong khai phá mẫu đồ thị con thường xuyên là không phẳng. Những nghiên cứu này góp phần giảm độ phức tạp đẳng cấu đồ thị con. Việc dùng ma trận kề mặc dù đơn giản không vay mượn cấu trúc tự nó đã có phát hiện đẳng cấu vì các đỉnh (và cạnh) có thể được liệt kê theo nhiều cách khác nhau [86]. Đối với quá trình kiểm tra đẳng cấu tuân theo chiến lược gắn mã nhất quán đảm bảo hai đồ thị bất kỳ được gán nhãn theo cùng một cách trong trật tự của đỉnh và cạnh (chiến lược gắn nhãn chuẩn). Một chiến lược gắn nhãn chuẩn xác định duy nhất một mã cho một đồ thị cho trước [72]. Gắn nhãn chuẩn tạo điều kiện kiểm tra đẳng cấu vì nó đảm bảo nếu một cặp đồ thị là đồng dạng thì mã chuẩn của chúng là giống nhau hoàn toàn [52]. Một cách đơn giản để sinh ra mã chuẩn là làm phẳng các ma trận kề bằng cách nối các cột hoặc hàng để xuất ra một mã bao gồm các số theo thứ tự quy định. Để giảm kết quả tính toán hoán vị các ma trận, gắn nhãn chuẩn được nén

và sử dụng lược đồ đỉnh bất biến [72] cho phép nội dung của ma trận kề được phân hoạch theo nhãn của đỉnh.

Một số phương pháp giảm không gian tìm kiếm khác tập trung vào phát hiện một tập con của tập tất cả các đồ thị con thường xuyên như khai phá đồ thị con thường xuyên đóng [89] hoặc khai phá đồ thị con thường xuyên cực đại [45], [81]. Mặc dù các phương pháp này có thể giới hạn một chút phạm vi nhưng vấn đề bùng nổ tổ hợp vẫn chưa thể giải quyết được và vẫn còn một số lượng lớn các đồ thị con thường xuyên đóng và đồ thị con thường xuyên cực đại được sinh ra.

Nghiên cứu sinh đề xuất một phương pháp khai phá mẫu đồ thị con thường xuyên với việc kiểm tra đẳng cấu đồ thị con trong thời gian đa thức với ràng buộc gán nhãn và thứ tự nhãn của cả đỉnh và cạnh trong tất cả đồ thị của cở sở dữ liệu đồ thị. Dựa trên phương pháp xác định mã duy nhất cho một đồ thị [72] để sinh ra đồ thị con ứng viên không trùng lặp và sử dụng tính chất hai cặp đồ thị đồng dạng thì mã chuẩn của chúng giống nhau hoàn toàn [52] cùng với sử dụng mô hình máy truy cập ngẫu nhiên [73] để kiểm tra đẳng cấu đồ thị con trong thời gian đa thức. Đề xuất của nghiên cứu sinh có ý nghĩa quan trọng trong vấn đề khai phá dữ liệu với các biểu diễn dữ liệu có cấu trúc phức tạp được chuyển đổi về biểu diễn dưới dạng đồ thị.

Hơn nữa, để giảm bớt số lượng mẫu đồ thị con thường xuyên được sinh ra, một khái niệm là khai phá mẫu đồ thị con đóng [89] và đồ thị con cực đại [45], [81] đã được đề xuất, nghiên cứu sinh sử dụng khai phá mẫu đồ thị con thường xuyên đóng nhằm giảm bớt số lượng ứng viên và các đồ thị con thường xuyên được sinh ra. Thuật toán mới do nghiên cứu sinh đề xuất cho việc khai phá các đồ thị con thường xuyên đóng dựa trên chiến lược nhãn chuẩn hóa, mô hình máy truy cập ngẫu nhiên (RAM) hoặc mô hình von Neumann [73] và cách tiếp cận Apriori. Bộ nhớ RAM sử dụng bộ nhớ truy cập ngẫu nhiên, qua đó vượt qua giới hạn của máy Turning sử dụng băng truy cập tuần tự. Bộ nhớ RAM có thể truy cập vào bất kỳ trường nào trong bộ nhớ của

chúng trong một bước để biết được ô nào truy cập và mỗi ô phải được chỉ định địa chỉ. Vấn đề đồ thị con đẳng cấu với chiến lược nhãn chuẩn hóa được xác định bằng cách tìm kiếm một phần tử trong một mảng theo chuỗi mã và có thể áp dụng tìm kiếm nhị phân trong RAM. Trong thuật toán mới, bài toán đồ thị con đẳng cấu được giải quyết trong thời gian đa thức so với giải quyết trong thời gian không đa thức trong các thuật toán trong [44], [46], [52], [88], [89]. Thêm vào đó nghiên cứu sinh cũng chỉ ra tính đúng đắn và độ phức tạp của thuật toán mới được đề xuất.

### 3.2.1 Ý tưởng đề xuất

Tính chất bài toán tìm tất cả các đồ thị con thường xuyên là có tồn tại một tính chất phản đơn điệu tức là tất cả các đồ thị con là thường xuyên thì đồ thị con của đồ thị con thường xuyên cũng là một đồ thị con thường xuyên và nếu một đồ thị con là thường xuyên thì tất cả các đồ thị cha của đồ thị con thường xuyên cũng là đồ thị con không thường xuyên. Hầu như tất cả các công trình nghiên cứu khai phá đồ thị con thường xuyên đều sử dụng tính chất này để tỉa các đồ thị con ứng viên thỏa mãn tính chất thường xuyên hay không trước khi sinh ra đồ thị con ứng viên mức cha của nó. Chẳng hạn như tìm tất cả các đồ thị con thường xuyên mức 2 rồi mới sinh ra các đồ thị con ứng viên mức 3, sau đó mới kiểm tra tính chất thường xuyên của các đồ thị con ứng viên mức 3 và loại đi các đồ thị con ứng viên mức 3 không thỏa mãn tính chất thường xuyên và tìm được tất cả các đồ thị con ứng viên mức 3. Thực hiện đệ quy công việc này đến khi không còn sinh ra các đồ thị con ứng viên thì dừng lại.

Đồ thị con ứng viên hay thường xuyên mức $k$ có thể là $k$ đỉnh hoặc $k$ cạnh tùy theo từng công trình nghiên cứu. Trong nghiên cứu của nghiên cứu sinh, sử dụng đồ thị con mức $k$ là có $k$ đỉnh. Để sinh ra một đồ thị con ứng viên mức $(k+1)$ đỉnh thì phải kết hợp hai đồ thị con thường xuyên mức $k$ đỉnh mà hai đồ thị con thường xuyên này phải có chung một đồ thị con thường xuyên mức $(k-1)$ đỉnh. Như vậy, đối với một tập các đồ thị con thường xuyên mức $k$ đỉnh ta sẽ sinh ra được một tập các đồ thị

con ứng viên mức $(k+1)$ đỉnh. Số lượng các đồ thị con ứng viên mức $(k+1)$ đỉnh này rất lớn và thuộc lớp hàm mũ giả sử là $2^n$. Vì vậy, khi so sánh một đồ thị con ứng viên mức $(k+1)$ đỉnh này có phải là đồ thị con đẳng cấu với một đồ thị $g_i \in \mathbb{GD}$, $\mathbb{GD}$ là cơ sở dữ liệu đồ thị giao tác, thì sẽ phải kiểm tra $2^n$ lần.

Nghiên cứu sinh sử dụng một mã chuẩn hóa cho mỗi đồ thị con ứng viên mức $(k+1)$ đỉnh sao cho các đồ thị con ứng viên mức $(k+1)$ đỉnh này có thể sắp xếp vào một mảng có thứ tự, sau đó sử dụng máy truy cập ngẫu nhiên và kỹ thuật tìm kiếm nhị phân để xác đinh một đồ thị con ứng viên mức $(k+1)$ có phải làm một đồ thị con của $g_i$ hay không, nếu có thì tăng biến đếm của nó lên một giá trị. Việc xác định này chính là kiểm tra đồ thị con đẳng cấu, và theo lý thuyết tìm kiếm nhị phân ta thấy công việc này được thực hiện với độ phức tạp $O(log_2 2^n) = O(n)$ Áp dụng cho tất cả các $g_i$ và xác định biến đếm của nó có thỏa mãn tính chất thường xuyên hay không. Nếu có thì đưa và danh sách đồ thị con thường xuyên mức $(k+1)$. Thực hiện đệ quy đến khi không sinh được thêm đồ thị con ứng viên mức cao hơn được nữa.

Ý tưởng thực hiện kiểm tra đẳng cấu trong thời gian đa thức đối với bài toán tìm tất cả các đồ thị con thường xuyên dựa trên các cấp bậc mỗi lần sinh ra các đồ thị con ứng viên và kiểm tra đẳng cấu đồ thị con. Đồ thị sẽ được biểu diễn duy nhất bằng bằng cách sử dụng một mã nhãn chuẩn hóa, có thứ tự tương tự MDFS-C [88] và CAM [44], [46], [52]. Biểu diễn duy nhất của đồ thị có mã CAM hoặc MDFS-C sẽ tránh tạo ra bản sao con nhân bản và có thể xây dựng các mảng được sắp xếp mà các đồ thị con chứa mã của MDFS-C hoặc CAM. Thêm vào đó, kiểm tra đồ thị con đẳng cấu có thể được sử dụng tìm kiếm nhị phân trong mô hình máy truy cập ngẫu nhiên. Xem xét vấn đề đồ thị con đẳng cấu là so khớp một cặp hai đồ thị con. Vấn đề này tương đương với so sánh một cặp chuỗi mã của biểu diễn MDFS-C hoặc CAM của hai đồ thị con. Độ phức tạp thời gian của tìm kiếm nhị phân là $O(logn)$ trong đó $n$ là kích thước đầu vào của mảng.

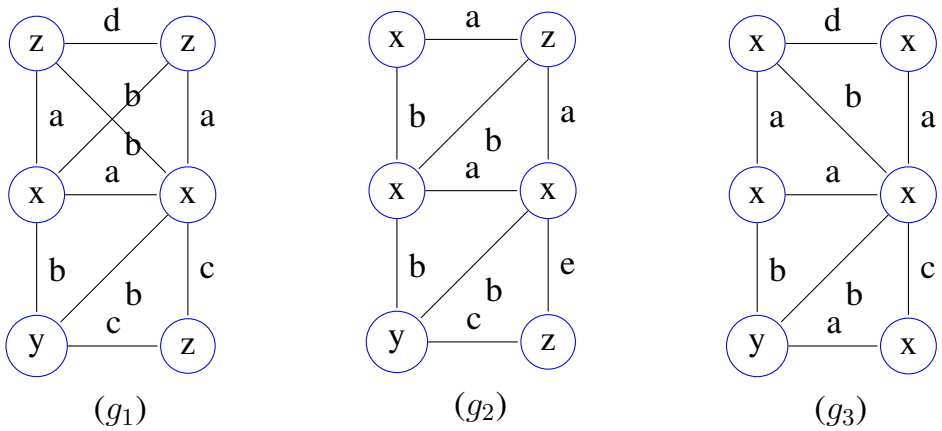
Thuật toán được gọi là Polynomial Subgraph Isomorphism Closed Frequency Subgraph Mining. Thuật toán sẽ dùng tiếp cận apriori-based và mô hình máy truy cập ngẫu nhiên. Theo cách tiếp cận apriori-based, tiến trình khai phá đồ thị con thường xuyên bắt đầu từ mức một sinh ra các ứng viên 2-đồ thị con của mỗi đồ thị $G_i$ trong cơ sở dữ liệu đồ thị $\mathbb{GD}$, kiểm tra đồ thị con đẳng cấu của các ứng viên này và đếm độ hỗ trợ để tìm ra các 2-đồ thị con thường xuyên của đồ thị $G_i$, tập 2-đồ thị con thường xuyên của cơ sở dữ liệu đồ thị $\mathbb{GD}$, tập 2-đồ thị con thường xuyên đóng của đồ thị $G_i$, tập 2-đồ thị con thường xuyên đóng của cơ sở dữ liệu đồ thị $\mathbb{GD}$ tương ứng các ký hiệu lần lượt là $FS_2^i$, $FS_2$, $CS_2^i$ và $CS_2$.

Tiến trình tiếp theo là một vòng lặp với $k \geqslant 3$ sẽ sinh ra tập $C_k^i$ của tập ứng viên k-đồ thị con (các đồ thị có k đỉnh và có cạnh nằm trong khoảng từ (k-1) đến (k(k-1)/2) cạnh) của đồ thị $G_i$ từ tập 2-đồ thị con thường xuyên $FS_2^i$ kết hợp với tập (k-1)-đồ thị con thường xuyên $FS_{k-1}^i$, kiểm tra đẳng cấu các ứng viên k-đồ thị con và đếm độ hỗ trợ của các ứng viên k-đồ thị con của $C_k^i$ để tìm tập k-đồ thị con thường xuyên của $G_i$, một tập k-đồ thị con thường xuyên của $\mathbb{GD}$, một tập k-đồ thị con thường xuyên đóng của $G_i$ và tập k-đồ thị con thường xuyên đóng của $\mathbb{GD}$ ký hiệu lần lượt là $FS_k^i$, $FS_k$, $CS_k^i$ and $CS_k$.

Trong tiến trình mức $k$ này, mọi k-đồ thị con của $CS_k^i$ và $CS_k$ được xây dựng với một danh sách liên kết chứa các (k-1)-đồ thị con của $FS_{k-1}^i$ và $FS_{k-1}$, thuật toán sẽ kiểm tra mỗi k-đồ thị con của $CS_k^i$ và $CS_k$ mà chứa k-đồ thị con chứa (k-1)-đồ thị con của $CS_{k-1}^i$ và $CS_{k-1}$ trong danh sách liên kết của chúng và loại bỏ những (k-1)-đồ thị con giống nhau trong $CS_{k-1}^i$ và $CS_{k-1}$. Tiến trình lặp sẽ dừng khi không có ứng viên k-đồ thị con nào được sinh ra. Hợp của các tập $CS_1, CS_2, ..., CS_k$ là tập các đồ thị con thường xuyên đóng.

**Định nghĩa 3.2.1.** Một *k-đồ thị con* của đồ thị $g$ là một đồ thị con $g' \subseteq g$ mà $|V_{g'}| = k$.

### 3.2.2 Nhãn chuẩn hóa




Hình 3.1: Một cơ sở dữ liệu đồ thị giao tác $\mathbb{GD}$

Cho một đồ thị $G$ với $n$ đỉnh, có thể biểu diễn đồ thị $G$ bằng một ma trận kề. Tuy nhiên, khi thay đổi thứ tự các đỉnh sẽ nhận được một ma trận kề khác cũng biểu diễn đồ thị $G$. Số lượng ma trận kề biểu diễn đồ thị $G$ chính là số lượng hoán vị tất cả các đỉnh của đồ thị $G$. Do vậy sẽ có $n!$ ma trận kề cùng biểu diễn đồ thị $G$. Trong một hệ thống khai phá dữ liệu, phải sử dụng $n!$ biến thể biểu diễn cho một đồ thị $G$ gây khó khăn cho quá trình khai phá dữ liệu. Theo đó, cần phải sử dụng một biểu diễn duy nhất cho một đồ thị thì mới tăng được hiệu quả khai phá. Trong các công trình nghiên cứu [44], [46], [52], [88] đã chỉ ra việc sử dụng biểu diễn duy nhất cho một đồ thị làm giảm thời gian thực hiện khai phá đồ thị con thường xuyên. Trong các công trình này đã chỉ ra hai phương pháp biểu diễn duy nhất cho một đồ thị như sau:

1. *Minimum DFS Code (M-DFSC)*: Mỗi cạnh của đồ thị trong mã DFS được biểu diễn bởi một bộ năm $(i, j, l_i, l_e, l_j)$ với $i$ và $j$ là định danh các đỉnh, $l_i$ và $l_j$ là nhãn cho các đỉnh tương ứng với định danh, và $l_e$ là nhãn của cạnh nối hai đỉnh. Dựa trên thứ tự được sắp xếp của đồ thị DFS, M-DFSC của đồ thị $g$ được định nghĩa là nhãn chuẩn của $g$ [88]. Mã DFS cho nhánh trái nhất và nhánh phải nhất của đồ thị $G$ trong ví dụ Hình 3.1(g1) tương ứng là $\{(0, 1, x, a, x), (1, 2, x, a, z), (2, 3, z, d, z)\}$ và $\{(0, 1, x, a, x), (1, 4, x, b, y), (4, 5, y, c, z)\}$.

2. *Canonical Adjacency Matrix (CAM)*: Cho ma trận kề $M$ của đồ thị $G$, một mã của $M$ là chuỗi ghép ma trận tam giác dưới của $M$ bao gồm cả các thành phần trong đường chéo. Từ những hoán vị khác nhau của tập các đỉnh sẽ cho ra các ma trận kề khác nhau, mã chuẩn (CAM) của $g$ được xác định là mã lớn nhất (hoặc nhỏ nhất) [44], [46], [52]. Mã CAM của đồ thị $G$ trong Hình 3.1(g1) có mã code(CAM(g1)) = xaxabzabdzbb00yc000cz.

### 3.2.3 Sinh tập ứng viên

gSpan [88] đã phát triển một cách hiệu quả để giảm tổng số nút cần được xem xét. Trong gSpan, hoạt động mở rộng chỉ được thực hiện cho các nút trên "con đường bên phải nhất" của một đồ thị. Với một đồ thị $G$ và một trong những cây tìm kiếm độ sâu đầu tiên của nó $T$, con đường bên phải nhất của $G$ đối với $T$ là đường dẫn bên phải của cây $T$. gSpan chỉ chọn một cây tìm kiếm độ sâu đầu tiên $T$ tạo ra dạng chuẩn hóa $G$ để mở rộng. gSpan mở rộng một cạnh sang phải hầu hết các đường dẫn để nhận (k+1)-đồ thị con từ k-đồ thị con (k-đồ thị con trong gSpan có nghĩa là đồ thị con có $k$ cạnh).

Thuật toán FFSM [44] sử dụng hai thủ tục FFSM_Extension và FFSM_Join để tạo ra ứng cử viên đồ thị con. FFSM_Join kết hợp hai k-đồ thị con để tạo ra (k+1)-đồ thị con nếu hai k-đồ thị con chia sẻ chung (k-1)-đồ thị con (k-đồ thị con trong FFSM có nghĩa là đồ thị con có $k$ cạnh) và FFSM_Join không sinh ra (k+1)-đồ thị con duy nhất từ hai k-đồ thị con. FFSM_Extension cải thiện hiệu quả gSpan bằng cách luôn luôn chọn một nút cố định duy nhất trong CAM và đính kèm một cạnh vừa được giới thiệu vào nó cùng với một nút bổ sung.

Trong thuật toán mới, PSI-CFSM, nghiên cứu sử dụng kỹ thuật liệt kê là một hoạt động mở rộng liên quan đến việc đề xuất một ứng viên (k+1)-đồ thị con $G$ từ k-đồ thị con $G_i$ bằng cách đưa thêm các cạnh (k-đồ thị con có nghĩa là đồ thị con có $k$ đỉnh).

Cạnh mới được giới thiệu có thể kết nối hai nút hiện có hoặc kết nối một nút hiện có và một nút được giới thiệu cùng với cạnh. Một cách đơn giản để thực hiện hoạt động mở rộng là giới thiệu mọi cạnh có thể cho mỗi nút trong một đồ thị $G$.

Ý tưởng của việc thực hiện là xác định tất cả các $FS_2^i$, với mọi đồ thị con đóng thường xuyên từ tập $CS_{k-1}^i$, xây dựng tập đồ thị con ứng viên $C_k^i$ bằng cách mở rộng đồ thị thêm một đỉnh lấy mỗi cạnh $e$ trong $FS_2^i$ tương ứng với đỉnh cần thêm cho mỗi đồ thị $G$ và thêm các cạnh này vào đồ thị $G$ nếu đồ thị $G$ không chứa cạnh $e$.

Việc sinh ứng viên này kết thúc nếu không có đỉnh nào được thêm vào mọi đồ thị trong $CS_{k-1}^i$. Phương pháp này rõ ràng có độ phức tạp thời gian đa thức cho tập các đỉnh và đỉnh có sẵn cho một đồ thị $G$, tương ứng. Số lượng các cạnh có thể được thêm được giới hạn bởi cận trên $|CS_{k-1}^i| \times |FS_2^i|$.

Thủ tục *Combine* [1] kết hợp tập đồ thị con thường xuyên mức $k-1$ là $FS_{k-1}^i$ với tập đồ thị con thường xuyên mức 2 là tập các cạnh thường xuyên còn lại chưa được ghép $FS_2^i$ để sinh ra tập đồ thị con ứng viên $C_k^i$. Đối với mỗi đồ thị con thường xuyên trong $FS_{k-1}^i$ chỉ có tối đa là $(k-1) * (k-2)/2$ cạnh nên cũng chỉ có thể ghép thêm tối đa $(k-1) * (k-2)/2$ đồ thị con thường xuyên của $FS_2^i$ còn lại chưa được ghép.

Trong các tập cơ sở dữ liệu giao tác thực tế như Chemical Compound thì sẽ có rất nhiều các cạnh giống nhau về nhãn, chẳng hạn như C-C, C-O, C-H, O-H nên đối với dạng dữ liệu có cấu trúc biểu diễn dạng đồ thị này có thể lược bỏ rất nhiều cạnh giống nhau về hình thức. Từ đó số lượng cạnh hay 2-đồ thị được ghép với $FS_{k-1}^i$ để được $CS_k^i$ sẽ nhỏ hơn $(k-1) * (k-2)/2$ nhiều lần. Đây là một giá trị hữu hạn do $k$ hữu hạn vì $k$ không thể vượt quá số cạnh trong đồ thị $g_i \in \mathbb{GD}$.

Do đó thủ tục *Combine* này có độ phức tạp thời gian đa thức.

**Procedure** Combine($F^i_{k-1}, F^i_2$)

**Đầu vào:** một tập $F^g_{k-1} \subseteq FS(g)$, một tập $F^g_2 \subseteq FS(g)$

**Đầu ra :** tập ứng viên k-đồ thị con của đồ thị $g$ ký hiệu $C^g_k$

```
1  C^i_k ← ∅;
2  foreach u ∈ F^g_{k-1} do
3      ng ← u;
4      tg ← diagonal(u);
5      foreach v ∈ F^g_2 do
6          ag ← diagonal(v) // ag = (x, y);
7          if ((x ∈ ag ∧ x ∈ tg ∧ y ∉ tg) ∨ (y ∈ ag ∧ y ∈ tg ∧ x ∉ tg)) then
8              thêm một dòng mới trong u;
9              đặt location_u(|diagonal(u)| + 1, col(x)) = location_v(2, 1);
10         else
11             if ((x ∈ ag ∧ x ∈ tg ∧ y ∈ tg) ∨ (y ∈ ag ∧ y ∈ tg ∧ x ∈ tg)) then
12                 đặt location_u(row(y), col(x)) = location_v(2, 1);
13             end
14         end
15     end
16     thêm ng vào C^i_k;
17 end
18 trả về C^i_k;
```

**Bổ đề 3.2.2.** Thủ tục $Combine(F^i_{k-1}, F^i_2)$ là đúng đắn.

*Chứng minh.* Nghiên cứu sinh chứng minh tính đúng đắn của thủ tục $Combine(F^i_{k-1}, F^i_2)$ theo quy nạp cho $k \geqslant 3$. Bước cơ bản $k = 3$, $Combine(F^i_2, F^i_2)$ sinh ra tập $C^i_3$ các ứng viên $3 - subgraph$ của đồ thị $g_i \in \mathbb{GD}$, $C^i_k$. Rõ ràng là cho $u, v$ là hai đồ thị

con thuộc $F_2^i$, $diagonal(u) = \{u_x, u_y\}$, $diagonal(v) = \{v_x, v_y\}$, nếu $diagonal(u) \neq diagonal(v)$ và $((u_x = v_x) \vee (u_x = v_y) \vee (u_y = v_x) \vee (u_y = v_y))$ thì sự kết hợp $u, v$ sinh ra các ứng viên $3-subgraph\ sg \subseteq g_i \in \mathbb{GD}$. Bước quy nạp: tại bước $k > 3$, giả sử rằng $Combine(F_{k-1}^i, F_2^i)$ là đúng và sinh ra $C_k^i$. Tập $C_k^i$ sau khi tỉa $c \in C_k^i \wedge sup_c < \sigma$ đạt được $FS_k^i$. Cần phải chứng minh rằng $Combine(F_k^i, F_2^i)$ là đúng và sinh ra $C_{k+1}^i$. Tại bước $k+1$, $Combine(F_k^i, F_2^i)$ sinh ra $C_{k+1}^i$, cho $sc \in C_{k+1}^i, sp \in C_k^i$, tìm được $diagonal(sc)$ và $diagonal(sp)$ và $|diagonal(sc)| - |diagonal(sp)| = 1$. Bỏ đi tất cả các cạnh chứa nút $nx = diagonal(sc) - diagonal(sp)$ đạt được đồ thị con $nc$, $nc \in F_k^i$. Theo giả thiết quy nạp $nc$ phải thuộc vào $C_k^i$ tập đồ thị con ứng viên k-subgraph. Mặc khác, $nc \in FS_k^i, FS_k^i \subseteq C_k^i \rightarrow nc \in C_k^i$. Do đó, tại bước $k+1$, $C_{k+1}^i$ là đúng với $Combine(F_k^i, F_2^i)$ và sau khi tỉa $rs \in C_{k+1}^i \wedge sup_r s < \sigma$ đạt được $FS_{k+1}^i$. □

**Bổ đề 3.2.3.** Thủ tục $Combine(F_{k-1}^i, F_2^i)$ được thực hiện trong thời gian đa thức.

*Chứng minh.* Cho $m$ là số lượng các cạnh của đồ thị con $g \in F_{k-1}^i$, $|diagonal(g)| = k-1$, $m \geqslant |diagonal(g)| - 1 = k-2$, $n$ là lực lượng của $F_2^i$ (mỗi đồ thị con trong $F_2^i$ chứa duy nhất một cạnh), $h$ là lực lượng của $F_{k-1}^i$, trước khi thêm một đỉnh vào đồ thị con (k-1)-subgraph $g$ để đạt được k-subgraph $g'$ thì số nút tối đa có thể được thêm vào $g$ là $|V_{g_i}| - |diagonal(g)|$. Theo biểu diễn CAM của đồ thị, số lượng cạnh của $g$ tối đa là $\sum_{i=1}^{k-2} i = \frac{(k-2) \times (k-2-1)}{2}$. Giả sử mỗi đồ thị con $g \in F_{k-1}^i$ có thể thêm tối đa số cạnh khi sinh ứng viên k-subgraph $g' \in C_k^i$ thì số cạnh có thể thêm vào $g$ là $(k-1)$. Do đó, thủ tục $Combine(F_{k-1}^i, F_2^i)$ có số bước tính toán tối đa là $h \times (|V_{g_i}| - (k-1)) \times (k-1)$ là một hàm đa thức. Trong thuật toán nghiên cứu sinh đề xuất hướng mục tiêu vào khai phá đồ thị con thường xuyên đóng, dùng thủ tục $Combine(CS_{k-1}^i, FS_2^i)$ để sinh tập ứng viên k-subgraphs $C_k^i$ và lực lượng của $C_k^i$ nhỏ hơn rất nhiều so với sử dụng $Combine(FS_{k-1}^i, FS_2^i)$. □

### 3.2.4 Kiểm tra đồ thị con đẳng cấu

Trong các thuật toán [44], [46], [52], [88] bằng cách sử dụng kiểm tra đẳng cấu đồ thị con theo trình cách tuần tự. Do đó, một ứng viên đồ thị con mới $g$ khi được tạo ra bởi phần mở rộng bên phải đường dẫn (gSpan) [88] hoặc kết nối (join) (FFSM) [44] sẽ được kiểm tra đồ thị con đẳng cấu với mỗi đồ thị $g' \subseteq g_i$ trong $\mathbb{GD}$. Quá trình kiểm tra đồ thị con đẳng cấu là so sánh MDFS-C hoặc CAM của một đồ thị con ứng viên $g$ với mỗi đồ thị con $g' \subseteq g_i \in \mathbb{GD}$ trong tập hợp đồ thị con rất lớn của đồ thị $g_i \in \mathbb{GD}$. Giả sử số đồ thị con của đồ thị $g_i \in \mathbb{GD}$ là $2^n$ thì quá trình này có $2^n$ bước so sánh. Có thể thấy rằng quá trình này vẫn chạy chậm. Do đó, bước kiểm tra đồ thị con đẳng cấu của các thuật toán FFSM, gSpan, CloseGraph có độ phức tạp thời gian thuộc lớp không đa thức (non-polynomial NP).

Trong thuật toán gSpan việc xác định đẳng cấu đồ thị con được thực hiện trên một cây gọi là "DFS Code Tree" (Hình 3.2) là một không gian tìm kiếm phân cấp. Trong cây không gian tìm kiếm phân cấp các đỉnh chứa các đồ thị con có $n$ cạnh là các con của các đỉnh chứa các đồ thị con có (n-1) cạnh tương ứng. Yan và Han [88] đã phát triển một thứ tự gọi là "lexicographic order" cho mỗi đỉnh trong không gian tìm kiếm sử dụng mã DFS (phương pháp gắn nhãn chuẩn hóa mới so với [46], [52] để ánh xạ một đồ thị tới một biểu diễn chuỗi duy nhất.

Theo sự phân tích trong [52] thì với mỗi một đồ thị sẽ có $n!$ biểu diễn đồ thị con bằng cách hoán vị thứ tự các đỉnh của đồ thị có $n$ đỉnh. Do đó một đồ thị sẽ có nhiều mã DFS trong cây DFS Code Tree. Một đồ thị được gán nhãn chuẩn hóa sẽ gán mã đầu tiên theo thứ tự tìm kiếm duyệt trước theo độ sâu của cây DFS Code Tree gọi là mã DFS tối thiểu. Bằng cách tỉa đi các đỉnh chứa các mã DFS không tối thiểu sẽ làm giảm kích thước của cây tìm kiếm kéo theo làm giảm không gian tìm kiếm.

Giả sử cho một cây DFS $T$, hai đỉnh $v_i$ và $v_j$ được quy định thứ tự $v_i < v_j$ nghĩa

là đỉnh $v_i$ được duyệt trước đỉnh $v_j$. Mỗi đỉnh trong cây DFS $T$ sẽ được gán một chỉ số từ 0 đến $n-1$ nếu đồ thị $g$ có $n$ đỉnh và $\forall i, j$: $v_i < v_j$ nếu và chỉ nếu $i < j$. Đỉnh $v_0$ là gốc và đỉnh $v_{n-1}$ là đỉnh phải nhất. Đường đi từ đỉnh gốc $v_0$ tới đỉnh phải nhất $v_{n-1}$ là đường phải nhất của cây DFS $T$. Tập chứa tất cả các cạnh trong cây DFS gọi là cạnh tới (forward edge). Tập chứa tất cả các cạnh không nằm trong cây DFS gọi là cạnh lùi (backward edge). Thứ tự trên cạnh tiến và cạnh lùi trong cây $T$ được định nghĩa là $E_{f,T}$ và $E_{b,T}$ tương ứng trong đó $E_{f,T} = \{e|\forall i, j, i < j, e = (v_i, v_j) \in E\}$ và $E_{b,T} = \{e|\forall i, j, i > j, e = (v_i, v_j) \in E\}$. Dựa trên các định nghĩa này để xác định hai thứ tự bộ phận $<_{f,T}$ và $<_{b,T}$ đối với hai cạnh $e_1 = (v_{i_1}, v_{j_1})$, $e_2 = (v_{i_2}, v_{j_2})$: (i) $e_1 <_{f,T} e_2$ nếu và chỉ nếu $j_1 < j_2$ với $\forall e_1, e_2 \in E_{f,T}$; (ii) với $\forall e_1, e_2 \in E_{b,T}$: $e_1 <_{b,T} e_2$ nếu và chỉ nếu $(i_1 < i_2)$ hoặc $((i_1 = i_2) \wedge (j_1 < j_2))$.

gSpan sử dụng gắn mã chuẩn hóa để giải quyết vấn đề đẳng cấu đồ thị con theo mã DFS nhỏ nhất. Nếu mã chuẩn hóa của hai đồ thị là bằng nhau thì hai đồ thị là đẳng cấu với nhau cũng giống với các công trình nghiên cứu của [16], [58], [84] dựa trên ý tưởng cơ bản của Nauty đến từ tính chất mã chuẩn hóa của hai đồ thị đẳng cấu là bằng nhau. Mỗi một hoán vị các đỉnh của một đồ thị có thể nhận được một chuỗi ký tự biểu diễn cho đồ thị đó. Giữa các chuỗi ký tự này thì một chuỗi thứ tự nhỏ nhất được lấy làm mã chuẩn của đồ thị. Có một phương pháp cổ điển là hệ thống gắn nhãn chuẩn chung có thể được xây dựng từ việc ghép các hàng hoặc các cột của ma trận kề của đồ thị [44, 52, 46].

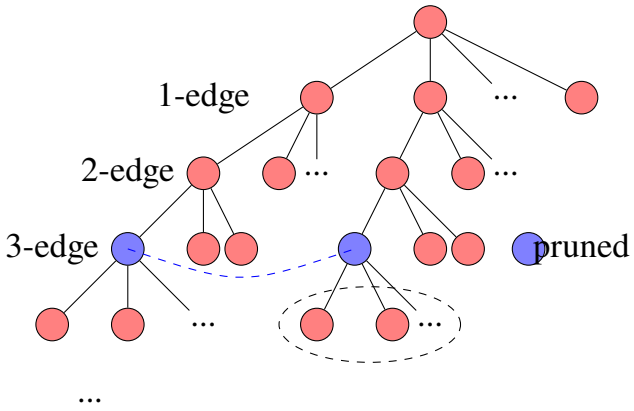
Cùng với việc xây dựng mã chuẩn, có thể dễ dàng chứng minh được các đồ thị đẳng cấu có một và chỉ một mã chuẩn. gSpan cũng đưa thêm thứ tự cho cả cạnh tiến và cạnh lùi ký hiệu $<_{bf,T}$ trên tập cạnh $E$ của cây $T$: $e_1 <_{bf,T} e_2$ nếu và chỉ nếu (i) $e_1 \in E_{b,T}$, $e_2 \in E_{f,T}$, $i_1 < j_2$ hoặc (ii) $e_1 \in E_{f,T}$, $e_2 \in E_{b,T}$, $j_1 \leqslant j_2$. Bằng việc kết hợp các thứ tự $<_{b,T}$, $<_{f,T}$ và $<_{bf,T}$ sẽ sinh ra thứ tự toàn thể trên một đồ thị $g$ của cây DFS $T$ ký hiệu $<_{E,T}$. Cho một mã DFS $\alpha = (a_0, a_1, ..., a_m)$, bất kỳ mã DFS hợp lệ $\beta = (a_0, a_1, ..., a_m, b)$ được gọi là một con và $\alpha$ được gọi là cha ký hiệu

$child(\alpha) = \{\beta | \forall \beta, \alpha = parent(\beta)\}$. Trong một cây "DFS Code Tree", mỗi đỉnh là biểu diễn một mã DFS, quan hệ giữa đỉnh cha và đỉnh con tuân theo quan hệ cha con $\alpha$ và $\beta$. Quan hệ giữa các anh em là nhất quán theo thứ tự DFS (lexicographic order) là thứ tự duyệt trước theo độ sâu của cây DFS Code Tree theo thứ tự DFS. Cây DFS Code Tree chứa tất cả các mã DFS nhỏ nhất của tất cả các đồ thị trong cơ sở dữ liệu đồ thị $\mathbb{GD}$. Cho hai mã DFS $\alpha$ và $\beta$ trong cây $T$, có một đường trực tiếp từ $\alpha$ tới $\beta$ thì $\alpha$ được gọi là tiền bối của $\beta$ ký hiệu $anc(\beta)$ và $\beta$ được gọi là hậu bối của $\alpha$ ký hiệu $des(\alpha)$.

Một đồ thị $g$ là thường xuyên thì tất cả các đồ thị con của $g$ cũng là thường xuyên. Nếu $g$ là không thường xuyên thì mọi đồ thị chứa đồ thị $g$ cũng là không thường xuyên. Có thể nói một cách tương tự là nếu một mã DFS là thường xuyên thì $\forall \beta \subset anc(\alpha)$, $\beta$ là thường xuyên, nếu $\alpha$ là không thường xuyên thì $\forall \beta \subset des(\alpha)$, $\beta$ là không thường xuyên. Cho một mã DFS $\beta$, nếu $\alpha = min(\beta), \alpha < \beta$ thì $D_\gamma = \{\varphi | \forall \varphi, \varphi < \gamma\}$, $\forall \sigma, \sigma \subset child(\beta)$ bất kỳ mã DFS được sinh ra bởi một cạnh thêm vào từ $\beta$, $min(\sigma) \subset D_\alpha \cup child(\alpha) \subseteq D_\beta$. Cho một đồ thị $g$ và các mã DFS của nó là $a_0, a_1, ..., a_n$, $\forall i, j \leqslant n$, $a_i \leqslant a_j$ (theo thứ tự DFS) và $a_0$ là mã DFS nhỏ nhất của $g$ thì cây DFS Code Tree sau khi tỉa tất cả các $a_i$, $1 \leqslant i \leqslant n$ và tất cả các hậu bối của nó trong các cây con vẫn thỏa mãn bảo toàn chứa tất cả các mã DFS của các đồ thị trong cơ sở dữ liệu đồ thị trên cây DFS Code Tree. Bằng việc tỉa bớt các mã DFS trong cây $T$ trước khi đưa vào cây DFS Code Tree sẽ đảm bảo cây DFS Code Tree sẽ chỉ chứa các ứng viên trong việc duyệt các đồ thị để xác định mức độ thường xuyên của nó.

Việc xác định đồ thị con đẳng cấu được chuyển thành ba bước gồm xây dựng các mã DFS, tỉa các mã DFS, rồi đưa dần các mã DFS của các cây vào trong cây DFS Code Tree. Trong ba bước này thì bước một là bước sinh ra nhiều nhất các mã DFS có thể so sánh nó với $n!$ hoán vị của các đồ thị đối với một đồ thị cho trước và sau khi đã sinh ra tất cả các mã DFS code nó mới có thể tìm được mã DFS code nhỏ nhất để đưa vào cây DFS Code Tree. Vì vậy việc xác định đồ thị con đẳng cấu của gSpan

chưa thực sự hiệu quả.

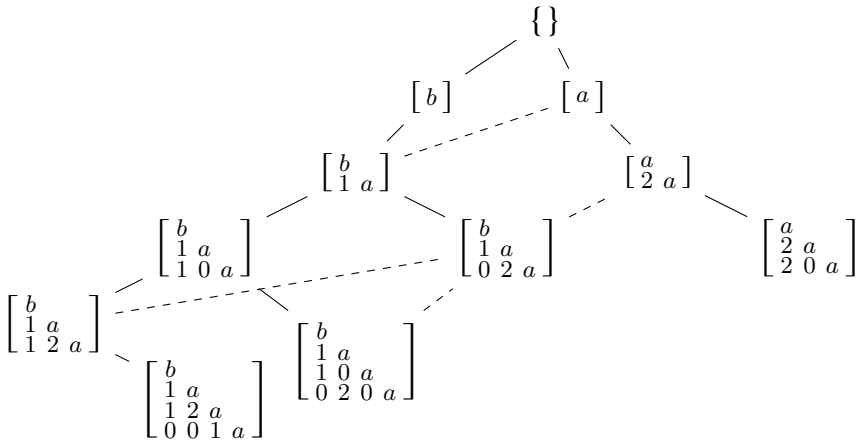



Hình 3.2: Cây đồ thị con thường xuyên: DFS Code Tree

Trong thuật toán FFSM [44] dùng ma trận tam giác dưới cùng với đường chéo của nó trong ma trận kề của đồ thị $g$ để biểu diễn mã cho đồ thị $g$. Sử dụng ký hiệu $m_{i,j}$ để đại diện cho mục ở hàng thứ $i$ và cột thứ $j$ của ma trận kề $M$ của đồ thị $g$, $0 < j \leqslant i \leqslant n$. Ma trận kề không phải là duy nhất ứng với một đồ thị đã cho, mỗi phần tử trên đường chéo của ma trận kề biểu diễn một đỉnh trong đồ thị, mỗi hoán vị của tập các đỉnh tương ứng với một ma trận kề khác. Sẽ có $n!$ ma trận kề khác nhau cho một đồ thị $n$ đỉnh. Cho một ma trận kề $M$, mỗi phần tử nằm trên đường chéo của $M$ là các phần tử biểu diễn đỉnh và mỗi thành phần không nằm trên đường chéo biểu diễn cạnh của đồ thị $g$. Thứ tự của các thành phần cạnh theo các vị trí tương đối trong mã của ma trận và cạnh đầu tiên của ma trận $M$ là cạnh trái nhất xuất hiện trong code(M) và cạnh cuối cùng là cạnh phải phất trong code(M).

FFSM sử dụng thứ tự (lexicographic order) trên các chuỗi ký tự xác định một thứ tự toàn thể giữa hai mã tùy ý $p$ và $q$. Cho một đồ thị $G$, một dạng chuẩn là mã lớn nhất trong tất cả các mã có thể. Ma trận kề $M$ cho ra dạng chuẩn được định nghĩa là ma trận kề chuẩn của $G$ (CAM - G's canonical adjacency matrix), chú trọng vào đồ thị vô hướng liên thông đơn. Thuộc tính quan trọng nhất của dạng chuẩn là "tiền tố" của dạng chuẩn cũng là lớn nhất.

Cho một đồ thị liên thông $G$ và một đồ thị con $H$ của $G$, cho $A$ và $B$ là các CAM của $G$ và $H$ tương ứng thì $code(A) \geqslant code(B)$. Cho một CAM $M$ của một đồ thị liên thông $G$ và $N$ là ma trận con của $M$ thì $N$ biểu diễn một đồ thị con liên thông của $G$. Cho một đồ thị liên thông $G$ cùng với CAM $M$ của nó, $N$ là ma trận con của $M$ và một đồ thị $H$ mà $N$ biểu diễn $H$ thì $N$ là CAM của $H$. Coi một ma trận rỗng là một ma trận con cực đại của bất kỳ ma trận nào có kích thước bằng 1 (ma trận biểu diễn của đồ thị chỉ có 1 đỉnh) thì có thể sắp xếp tất cả CAM của các đồ thị con liên thông của đồ thị $G$ vào trong một cây có gốc: (i) đỉnh gốc của cây là ma trận rỗng, (ii) mỗi đỉnh của cây là một đồ thị con liên thông phân biệt của $G$ được biểu diễn bởi CAM của nó, (iii) cho một đỉnh không phải là gốc với CAM $M$, cha của nó là một đồ thị được biểu diễn bởi ma trận con cực đại của $M$. Sau đó liệt kê tất cả các đồ thị con của tất cả các đồ thị trong cơ sở dữ liệu đồ thị $\mathbb{GD}$ trong một cây gọi là cây "CAM Tree" (Hình 3.3).



Hình 3.3: Cây đồ thị con thường xuyên: CAM Tree

FFSM sử dụng hai phương thức để sinh ra một đồ thị con (k+1)-cạnh từ đồ thị con k-cạnh bằng toán tử *kết nối* và *mở rộng*. Toán tử kết nối sẽ ghép hai đồ thị con thường xuyên k-cạnh $G_1$ và $G_2$ để được đồ thị con ứng viên (k+1)-cạnh $G$ với điều kiện $G_1$ và $G_2$ phải có chung đồ thị con (k-1)-cạnh. Toán tử kết hợp tốn nhiều thời gian và cũng sinh nhiều đồ thị ứng viên và các đồ thị ứng viên có thể dư thừa bởi các toán tử kết hợp khác nhau. Một cách khác để có đồ thị con ứng viên mới là thực hiện toán tử

mở rộng một đồ thị con thường xuyên. Toán tử mở rộng sẽ cho ra một đồ thị con ứng viên (k+1)-cạnh từ một đồ thị con thường xuyên k-cạnh bằng cách thêm một cạnh vào $G$ có thể sẽ kéo theo việc có thêm đỉnh mới hoặc không. Toán tử mở rộng cũng tốn nhiều thời gian thực hiện vì có nhiều đỉnh của đồ thị mà việc thêm một cạnh tới có thể sẽ được thêm vào đồ thị $G$.

Để giảm chi phí thời gian thực hiện thì FFSM sử dụng ma trận kề chuẩn con tối ưu dựa trên hai toán tử FFSM_Join và FFSM_Extension. Sử dụng CAM Tree và hai toán tử kết hợp và mở rộng với tối ưu ma trận kề chuẩn con sẽ liệt kê được hết tất cả các đồ thị con của cơ sở dữ liệu đồ thị $\mathbb{GD}$. Cho một đồ thị $G$, một ma trận kề chuẩn tối ưu con (supoptimal CAM) của $G$ là một ma trận kề $M$ của $G$ mà ma trận con cực đại $N$ là CAM của đồ thị mà $N$ biểu diễn. Cho một đồ thị $G$, cho $C_{k-1}$ ($C_k$) là tập các suboptimal CAM của tất cả các đồ thị con (k-1)-đỉnh (k-đỉnh) của $G$ ($k \geqslant 3$). Tất cả đồ thị con trong tập $C_k$ có thể được liệt kê không nhập nhằng bằng việc kết hợp hai thành viên của $C_{k-1}$ hoặc bằng mở rộng một thành viên của $C_{k-1}$. Như vậy, có thể mở rộng CAM Tree cho một tập hợp các đồ thị thay cho một đồ thị vô hướng liên thông đơn.

Một cây CAM Tree duy nhất có thể được xây dựng cho cả một cơ sở dữ liệu đồ thị sẽ làm giảm không gian và thời gian tính toán các đồ thị con thường xuyên. Quá trình duyệt cây CAM Tree sẽ phát hiện các đồ thị con của cơ sở dữ liệu đồ thị và với mỗi đồ thị con, độ hỗ trợ của nó có thể được quyết định bởi một bước quét tuyến tính cơ sở dữ liệu đồ thị và mức thường xuyên của một đồ thị con sẽ được ghi nhận vào kết quả. Có thể thấy được, việc kiểm tra đẳng cấu của FFSM sẽ phụ thuộc vào FFSM_Join, FFSM_Extension và duyệt cây CAM Tree duy nhất cho cả cơ sở dữ liệu đồ thị $\mathbb{GD}$. Mỗi một đồ thị con ứng viện được sinh ra sẽ phải duyệt lại hoàn toàn cây CAM Tree để xác định vị trí đặt cây và xác định độ hỗ trợ của đồ thị con ứng viên được đưa ra để tránh phải tìm đồ thị con đẳng cấu mà chỉ cần duyệt lại cây tìm tất cả các đồ thị con có độ hỗ trợ lớn hơn ngưỡng cho phép để kết luận các đồ thị con thường xuyên. Khi

các đồ thị con ứng viên có số lượng cạnh lớn thì việc duyệt lại cây để xác định vị trí đặt các đồ thị con ứng viên này sẽ tiêu tốn nhiều thời gian vì phải duyệt lại cây CAM Tree qua các mức đồ thị con trước đó nhiều lần nên không thể đạt được độ phức tạp thời gian đa thức.

Thủ tục *BinarySearch* [1] được thực hiện trên mô hình máy truy cập ngẫu nhiên có độ phức tạp thời gian đa thức.

**Procedure** BinarySearch(L,x,first,last)

**Đầu vào:** Mảng $L[first, last]$ và giá trị $x$

**Đầu ra :** $-1$ nếu $x \notin L$, hoặc $i, 0 \leqslant i < n$ nếu $L[i] = x$

```
1  if first > last then
2      trả về -1;
3  else
4      middle ← ⌈(first+last)/2⌉;
5      if L[middle] = x then
6          trả về middle;
7      else
8          if L[middle] < x then
9              trả về BinarySearch(L, x, middle + 1, last);
10         else
11             trả về BinarySearch(L, x, first, middle - 1);
12         end
13     end
14 end
```

Thuật toán PSI-CFSM cải tiến các bước kiểm tra đẳng cấu đồ thị con bằng cách sử dụng kiểm tra đẳng cấu đồ thị con theo tìm kiếm nhị phân trong mô hình máy truy cập ngẫu nhiên. Trong lý thuyết độ phức tạp tính toán, sự phức tạp về thời gian của

tìm kiếm nhị phân là $O(logn)$ trong đó $n$ là số lượng ứng viên đồ thị con. Giả sử lực lượng của các đồ thị con ứng viên là $2^n$ thì số bước của phép kiểm tra đẳng cấu đồ thị con bằng cách tìm kiếm nhị phân trên mô hình máy truy cập ngẫu nhiên là $log_2 2^n = n$ và độ phức tạp của thời gian là $O(n)$.

**Bổ đề 3.2.4.** Độ phức tạp tính toán của thủ tục BinarySearch là $O(log_2 n)$.

*Chứng minh.* Cho $T(n)$ là số bước tính toán của thủ tục $BinarySearch$ cần thực hiện với kích thước đầu vào là $n$. Tại bước $n = 0$ thì $T(0) = c'$, với $c'$ là một hằng số và $|L| = 0$, thủ tục thực hiện số bước tính toán là một hằng số. Tại bước $n > 0$, thủ tục thực hiện một hằng số $c$ các bước tính toán để tìm phần tử nằm chính giữa $L$, so sánh phần tử đó với $x$ và xác định nửa vùng bên trái hay bên phải của mảng $L$ để thực hiện đệ quy. Giả sử cả hai nửa vùng của mảng $L$ có cùng kích cỡ, $\frac{n-1}{2}$. Do đó, số bước tính toán mà $BinarySearch$ thực hiện khi $n > 0$ là $T(n) = c + T(\frac{n-1}{2})$. Công thức được tái diễn như sau:

$$T(n) = \begin{cases} c', & \text{với } n = 0 \\ c + T\left(\frac{n-1}{2}\right), & \text{nếu } n > 0 \end{cases}$$

Với $n > 0$:

$$T\left(\frac{n-1}{2}\right) = c + T\left(\frac{\frac{n-1}{2}-1}{2}\right) = c + T\left(\frac{n-1-2}{2^2}\right) = c + T\left(\frac{n-2^0-2^1}{2^2}\right)$$

$$T\left(\frac{n-2^0-2^1-...2^k}{2^{k+1}}\right) = c + T\left(\frac{n-2^0-2^1-...2^{k+1}}{2^{k+2}}\right) \tag{3.1}$$

Thủ tục dừng khi đối của của hàm $T$ bằng $0$ nghĩa là $T(n) =? + T(0)$:

$$\frac{n-2^0-2^1-...2^{k+1}}{2^{k+2}} = 0 \tag{3.2}$$

$$T(n) = c + c + ... + c + T\left(\frac{n-2^0-2^1-...2^{k+1}}{2^{k+2}}\right) \tag{3.3}$$

Từ tập phương trình trong triển khai (3.1), có $k+2$ phương trình, số lượng các hằng số $c$ trong phương trình (3.3) là $k+2$ và vì thế nên $T(n) = (k+2)c + c'$, theo phương trình (3.2) thì:

$$n = 2^0 + 2^1 + ... + 2^{k+1} = \sum_{i=0}^{k+1} 2^k = 2^{k+1} - 1 \tag{3.4}$$

Do vậy, lấy logarit theo cơ số 2 của cả hai vế trái và phải của phương trình (3.4), đạt được $k+2 = log_2(n+1)$. Do vậy, $T(n) = clog_2(n+1) + c'$. Cuối cùng kết luận rằng $T(n) = O(log_2 n)$. □

**Bổ đề 3.2.5.** Thủ tục $BinarySearch(L, x, first, last)$ là đúng đắn.

*Chứng minh.* Cần phải chứng minh rằng $\forall n \geqslant 0$, $BinarySearch(L, x, first, last)$ trả về một vùng trong mảng được sắp xếp thứ tự $L$ với $0 \leqslant first \leqslant last \leqslant |L|$ nếu giá trị $x$ trong mảng $L$ được sắp xếp thứ tự.

Bước cơ bản: tại bước $n = 0$, mảng được sắp xếp thứ tự $L$ chứa vùng từ chỉ số 0 đến chỉ số $|L|-1$, $BinarySearch(L, x, 0, |L|-1)$ trả về vùng $[0, |L|-1] \subseteq [0, |L|-1]$ để tìm giá trị $x$. Rõ ràng, giá trị $x$ nằm trong vùng $[0, |L|-1]$ của $L$, $L[0] \leqslant x \leqslant L[|L|-1]$

Bước quy nạp: giả sử tại bước $n \geqslant 0$ và $BinarySearch(L, x, first_n, last_n)$ trả về vùng $[first_n, last_n] \subseteq [0, |L|-1]$ để tìm giá trị $x$, $L[first_n] \leqslant x \leqslant L[last_n]$

Cần chứng minh rằng tại bước $n+1$, $BinarySearch(L, x, first_{(n+1)}, last_{(n+1)})$ phải trả về vùng $[first_{(n+1)}, last_{(n+1)}] \subseteq [0, |L|-1]$ để tìm giá trị $x$, $L[first_{(n+1)}] \leqslant x \leqslant L[last_{(n+1)}]$

$BinarySearch(L, x, first_{(n+1)}, last_{(n+1)})$ trả về vùng $[first_{(n+1)}, last_{(n+1)}]$:

$$L[first_n] \leqslant x \leqslant L[last_n] \text{ ( } hypothesis\ induction \text{ )} \tag{3.5}$$

Trường hợp 1:

$$L\left[\left[\frac{first_n + last_n}{2}\right]\right] < x \tag{3.6}$$

$$[first_{(n+1)}, last_{(n+1)}] = \left[\left[\frac{first_n + last_n}{2}\right] + 1, last_n\right].$$

Theo (3.6) (3.5) suy ra $L\left[\left[\frac{first_n + last_n}{2}\right] + 1\right] \leqslant x \leqslant L[last_n]$

$$L[first_{(n+1)}] \leqslant x \leqslant L[last_{(n+1)}] \tag{3.7}$$

Trường hợp 2:

$$L\left[\left[\frac{first_n + last_n}{2}\right]\right] > x \tag{3.8}$$

$$[first_{(n+1)}, last_{(n+1)}] = \left[first_n, \left[\frac{first_n + last_n}{2}\right] - 1\right].$$

Theo (3.8) (3.5) suy ra $L[first_n] \leqslant x \leqslant L\left[\left[\frac{first_n + last_n}{2}\right] - 1\right]$

$$L[first_{(n+1)}] \leqslant x \leqslant L[last_{(n+1)}] \tag{3.9}$$

Từ các phương trình (3.7), (3.9) bổ đề được chứng minh. □

---

**Procedure** TestIsomorphism($g \in C_k^j, C_k^i$)

---

**Đầu vào:** $g \in C_k^j, C_k^i$

**Đầu ra :** $true \vee false$

```
1 b ← BinarySearch({code(CAM(g' ∈ C_k^i))}, code(CAM(g)), 0, |C_k^i|);
2 if b > 0 then
3 |   trả về true;
4 else
5 |   trả về false;
6 end
```

---

**Bổ đề 3.2.6.** [1] Độ phức tạp tính toán của TestIsomorphism là $O(log_2|C_k^i|)$

*Chứng minh.* Điều này là hiển nhiên theo bổ đề 3.2.4. □

**Bổ đề 3.2.7.** [1] Thủ tục $TestIsomorphism(g \in C_k^j, C_k^i)$ là đúng đắn.

*Chứng minh.* Điều này là hiển nhiên theo bổ đề 3.2.5. □

### 3.2.5 Thuật toán PSI-CFSM

**Algorithm 7:** PSI-CFSM($\mathbb{GD}$, $\sigma$ = min_sup)

**Đầu vào:** Cơ sở dữ liệu đồ thị $\mathbb{GD}$, $\sigma$ = min_sup

**Đầu ra :** $CS_2, CS_3, ..., CS_k$, các tập đồ thị con thường xuyên đóng theo mức

1 Xây dựng mảng có thứ tự theo code(CAM) của $C_2^i$;
2 **foreach** $u \in C_2^i$ **do**
3  TestIsomorphism($u$, $C_2^j$) và tìm $sup_u \geqslant \sigma$ để đặt $u$ vào trong $FS_2^i$, $FS_2^D$, $CS_2^i$ và $CS_2$;
4 **end**
5 $k \leftarrow 3$;
6 **while** *Combine($\forall CS_{k-1}^i$, $\forall FS_2^i$) is not null* **do**
7  Xây dựng mảng có thứ tự theo code(CAM) của $C_k^i$;
8  **foreach** $u \in C_k^i$ **do**
9  TestIsomorphism($u$, $C_k^j$) và tìm $sup_u \geqslant \sigma$ để đặt $u$ vào trong $CS_k^i$ và $CS_k$;
10  Kiểm tra $v \in CS_{k-1}^i$ nếu $sup_v = sup_u$ thì xóa $v$ khỏi $CS_{k-1}$;
11  **end**
12  $k \leftarrow k + 1$;
13 **end**

Trong thuật toán PSI-CFSM [1], bước đầu tiên là xây dựng mảng được sắp xếp thứ tự theo trật tự của mã CAM của các đồ thị con với 2 đỉnh (chỉ có một cạnh) 2-subgraph của đồ thị $G_i$ trong cơ sở dữ liệu đồ thị $\mathbb{GD}$. Mảng được sắp xếp thứ tự này ký hiệu là $C_2^i$, $C_2 = \{C_2^i\}$. Với mỗi phần tử $u$ trong $C_2^i$, so sánh $codeCAM(u)$ với $codeCAM(v), v \in \{C_2^j = C_2 - C_2^i\}$. Nếu $code(CAM(u)) = code(CAM(v))$ thì tăng độ hỗ trợ của $u$ lên 1. Nếu $sup_u \geqslant \sigma$ thì đặt $u$ vào trong $FS_2$, $FS_2^i$. $FS_2$ ($FS_2^D$) là tập các đồ thị con thường xuyên 2-subgraphs của cơ sở dữ liệu đồ thị $\mathbb{GD}$ và $FS_2^i$ là tập các đồ thị con thường xuyên 2-subgraphs của đồ thị $G_i \in \mathbb{GD}$. Xây dựng một vòng lặp với $k \geqslant 3$ để tính $C_k^i, FS_k, FS_k^i, CS_k, CS_k^i$ dựa trên thuật toán PSI-CFSM.

**Bổ đề 3.2.8.** [1] Thuật toán PSI-CFSM là đúng đắn.

*Chứng minh.* Cần chứng minh PSI-CFSM sinh ra đúng và đủ tất cả các đồ thị con thường xuyên đóng của tất cả các đồ thị trong cơ sở dữ liệu đồ thị $\mathbb{GD}$. Chứng minh quy nạp với $k \geqslant 2$, cần phải chỉ ra rằng $FS_k$ được tính bởi thuật toán này trùng với tập đồ thị con thường xuyên k-subgraphs (đồ thị con thường xuyên mức $k$). Ở bước cơ bản, khởi đầu quy nạp $k = 2$, tương ứng với $FS_2$ là tập tất cả các đồ thị con thường xuyên 2-subgraphs (có 2 đỉnh và 1 cạnh) rất dễ dàng kiểm tra. Bước quy nạp, giả sử ở bước $k$, $FS_{k-1}$ là tập các đồ thị con thường xuyên $k-1$ đỉnh ($(k-1-subgraph)$), $FS_{k-1}$ trùng với tập đồ thị con thường xuyên mức $k-1$. Số lượng tối đa các cạnh của đồ thị con với $k-1$ đỉnh là $k(k-1)/2$. Cần chứng minh rằng $FS_k$ trùng với tập đồ thị con thường xuyên mức $k$. Ở bước quy nạp, đủ để chứng minh rằng cho một đồ thị con thường xuyên bất kỳ $X$ mức $k$, $X$ chắc chắn nằm trong $FS_k$. Do vậy, $X$ cũng phải là một thành viên của $FS_k^i$ của các đồ thị con k-subgraphs của một đồ thị $g_i$ trong cơ sở dữ liệu $\mathbb{GD}$. $FS_k^i$ đạt được bởi bước tỉa loại bỏ các ứng viên k-subgraphs $r \in C_k^i \wedge sup_r < \sigma$ với $C_k^i$ là kết quả của $Combine(CS_{k-1}^i, F_2^i)$. Theo bổ đề 3.2.2 thì $Combine(CS_{k-1}^i, FS_2^i)$ là đúng đắn. Do đó, $FS_k$ chứa $X$ và trùng với tập đồ thị con thường xuyên mức $k$. □

Xem xét về độ phức tạp tính toán của thuật toán PSI-CFSM. Giả sử rằng số lượng đồ thị trong cơ sở dữ liệu đồ thị $\mathbb{GD}$ (kích thước của cơ sở dữ liệu) là $n$ và mỗi bước tính toán là một hằng số 1. Dòng 1 của thuật toán số lượng bước tính toán là tổng số cạnh của tất cả các đồ thị trong cơ sở dữ liệu đồ thị $\mathbb{GD}$

$$\left(\sum_{g_i \in \mathbb{GD}} |E_{g_i}|\right)$$

Dòng 2 và 3, số lượng bước tính toán là

$$\left(\sum_{g_i \in \mathbb{GD}} |E_{g_i}|\right) \times \left(\sum_{g_i \in \mathbb{GD}} log_2|E_{g_i}|\right)$$

Từ dòng 4 đến dòng 9 là vòng lặp có $k$ bước. Dòng 4 thực hiện thuật toán Combine có số bước tính toán là:

$$|CS^i_{k-1}| \times (|V_{g_i}| - (k-1)) \times (k-1)$$

Dòng 5 số bước tính toán là:

$$|CS^i_{k-1}| \times (|V_{g_i}| - (k-1)) \times (k-1) \times \left(\sum_{g_i \in \mathbb{GD}} |C^i_k|\right)$$

Tương tự như tính toán với 2-đồ thị con, số bước tính toán của các dòng 5,6,7,8 và 9 là:

$$|CS^i_{k-1}| \times (|V_{g_i}| - (k-1)) \times (k-1) \times \left(\sum_{g_i \in \mathbb{GD}} |C^i_k|\right) \times \left(\sum_{g_i \in \mathbb{GD}} log_2|C^i_k|\right)$$

Giả sử rằng mọi đồ thị $g_i$ trong cơ sở dữ liệu đồ thị $\mathbb{GD}$ đều có $|V_{g_i}| \geqslant 3$ thì số bước tính toán từ dòng 1 đến dòng 3 là nhỏ hơn rất nhiều số bước tính toán từ dòng 4 đến dòng 9. Theo đó tổng số bước tính toán của thuật toán PSI-CFSM là:

$$\sum_{k=1}^{max(|V_{g_i}|)} |CS^i_{k-1}| \times (|V_{g_i}| - (k-1)) \times (k-1) \times \left(\sum_{g_i \in \mathbb{GD}} |C^i_k|\right) \times \left(\sum_{g_i \in \mathbb{GD}} log_2|C^i_k|\right)$$

## 3.3 Phân loại đa nhãn cho đồ thị

Nhiều lĩnh vực trong cuộc sống đòi hỏi các đối tượng phải được gán nhiều nhãn như ảnh, nhạc, gen, web. Không thể phủ nhận vai trò của phân loại đa nhãn trong việc giải quyết nhiều vấn đề quan trọng trong cuộc sống hiện đại. Phân loại đa nhãn giải quyết vấn đề gán một tập các nhãn cho mọi đối tượng trong một tập hợp dữ liệu. Tức là, mỗi đối tượng trong một tập dữ liệu có thể được gán đồng thời nhiều nhãn. Ví dụ về phân loại đa nhãn cảm xúc người nghe với các bài hát [82]. Vấn đề phân loại đơn nhãn là loại trừ lẫn nhau các nhãn. Cho $X$ là miền của các đối tượng và $Y$ là tập các nhãn, $H$ tập các hàm hàm phân loại cho $X \rightarrow Y$. Mục tiêu là tìm hàm phân loại $h \in H$ có khả năng tối đa $h(x) = y$ với $y \in Y$ là nhãn xác thực của $x$. Với phân loại đa nhãn, các lớp cơ bản là không loại trừ lẫn nhau có thể chồng đè lên nhau. Cho $B$ là tập các véctơ nhị phân có độ dài $|Y|$. $H$ là tập các hàm phân loại của $X \rightarrow B$. Mục tiêu là tìm hàm phân loại $h \in H$ mà tối thiểu một khoảng cách (ví dụ Hamming) giữa $h(x)$ và $b_x$ cho một đối tượng mới $x$. Trong một phương pháp xác suất, mục tiêu phân loại đối tượng $x$ là để tìm một hoặc nhiều nhãn lớp cơ sở trong một tập nhãn $C$ với một ngưỡng $T$ mà $P(c|x) > T, \forall c \in C$. Thông thường nhất, các tiếp cận hợp nhất đơn giản, như lấy số đông, lớn nhất và trung bình được sử dụng.

Lý thuyết Dempster Shafer là một khung hợp nhất các hàm phân loại dựa trên luật của Dempster tăng độ chính xác trong phân lớp. Áp dụng lý thuyết Dempster Shafer [22] để phân loại đa nhãn cho đồ thị tăng độ chính xác trong phân loại và giảm độ phức tạp. Denoeux đã giới thiệu phân loại đa nhãn áp dụng lý thuyết Dempster như [23], [24], [25] đã đề xuất một phương pháp đề giảm độ phức tạp tính toán trong thực hiện và kết hợp các hàm khối, khi các hàm niềm tin được xác định trên một tập con phù hợp của *khung phân biệt* được kết hợp với cấu trúc dàn giao. Phương pháp này áp dụng cho phân loại đa nhãn dựa trên hàm phân loại k-láng giềng gần nhất (KNN) minh chứng [24]. Theo như Denoeux [24] thì áp dụng lý thuyết Dempster Shafer cho

phân loại đa nhãn sử dụng phương pháp kNN, với mỗi thành phần của tập $k$ các láng giềng sẽ lấy được một hàm khối và theo luật Dempster hợp nhất $k$ hàm khối này để được duy nhất một hàm khối cho đối tượng mới $x$. Tuy nhiên, ở đây chỉ áp dụng lý thuyết Dempster Shafer cho phân loại đa nhãn cho tập dữ liệu các đối tượng có một biểu diễn véctơ. Dựa vào biểu diễn véctơ này, chúng ta có thể tìm được tập $k$ láng giềng gần nhất. Nhưng với dữ liệu đồ thị thì có nhiều cách để tìm được tập $k$ láng giềng bằng các phương pháp sử dụng độ đo khác nhau như *đẳng cấu đồ thị con*, *đồ thị con chung lớn nhất*, khoảng cách Hausdorff, các *đỉnh của hai đồ thị gần nhau*, các *cạnh của hai đồ thị gần nhau*, *các nhân đường đi ngắn nhất*, *khoảng cách sửa đổi đồ thị*. Các phương pháp này đều đưa ra phương pháp xác định để so sánh tìm được $k$ đồ thị láng giềng gần với đồ thị được xét. Tuy nhiên, mục tiêu của chúng ta cần tìm $k$ đồ thị láng giềng mà có tập nhãn xấp xỉ với $k$ tập nhãn của $k$ đồ thị láng giềng. Như vậy, ở đây phải có một mối liên hệ giữa tập nhãn và tập đồ thị. Mối liên hệ đó chính là chìa khóa mở ra việc đi tìm tập $k$ láng giềng. Theo đó, các phương pháp độ đo như đồ thị con chung lớn nhất của Hausdorff hay khoảng cách chỉnh sửa đồ thị, các nhân đường đi ngắn nhất không thể áp dụng được cho phân loại đa nhãn đồ thị dùng lý thuyết Dempster Shafer. Theo như [50] mối quan hệ giữa tập nhãn và tập đồ thị chính là tập các đồ thị con của đồ thị với tập các nhãn. [50] coi việc xác định các đặc trưng phù hợp cho phân loại đa nhãn là tìm ra tập các đồ thị con và coi tập đồ thị con là lựa chọn đặc trưng. [62] đã đưa ra một phương pháp xây dựng dàn giao khái niệm cho đồ thị áp dụng vào phân loại đồ thị nhưng chưa áp dụng cho phân loại đa nhãn cho đồ thị. Từ dàn giao khái niệm này, mà tính chất của đồ thị được thể hiện bởi các đồ thị con của nó, nghiên cứu sinh xây dựng dàn giao khái niệm áp dụng vào phân loại đa nhãn đồ thị theo lý thuyết Dempster Shafer, dàn giao khái niệm thỏa mãn mối quan hệ giữa tập nhãn và tập đồ thị làm độ đo tương tự trong quá trình tìm tập $k$ láng giềng của một đồ thị mới để xác định tập nhãn cho đồ thị mới cần phân loại đa nhãn.

Phân loại đa nhãn áp dụng cho nhiều lĩnh vực, tuy nhiên đối với dữ liệu biểu diễn

dạng đồ thị thì phân loại đa nhãn gặp nhiều khó khăn do đồ thị khó có thể véctơ hóa như các dạng biểu diễn dữ liệu khác. Tùy thuộc vào từng yêu cầu của từng vấn đề cụ thể sẽ có tương ứng cách biến đổi dữ liệu đồ thị sang biểu diễn véctơ. Do vậy sẽ gặp nhiều khó khăn trong quá trình xác định độ tương tự giữa hai đồ thị. Nghiên cứu sinh đã nghiên cứu và xây dựng thành công dàn giao khái niệm làm cơ sở để xác định độ đo tương tự giữa hai đồ thị. Từ đó có thể áp dụng phương pháp $k$ láng giềng gần nhất để tìm tập nhãn cho một đồ thị mới dựa trên tập nhãn của $k$ đồ thị láng giềng có độ đo gần nhất với nó. Để xây dựng dàn giao khái niệm, nghiên cứu sinh sử dụng thuật toán khai phá đồ thị con thường xuyên đóng PSI-CFSM để xác định tập đồ thị con thường xuyên đóng của mỗi đồ thị $g_i \in \mathbb{GD}$ do nó giải quyết được bài toán con là xác định đẳng cấu đồ thị con trong thời gian đa thức. Dựa vào các đồ thị con thường xuyên đóng này làm đặc trưng cho mỗi đồ thị và xây dựng khái niệm chính thức, quan hệ cha con giữa hai khái niệm chính thức và hình thành nên dàn giao khái niệm. Lý thuyết Dempster-Shafer đặc biệt sử dụng trong việc điều khiển các dữ liệu không chắc chắn hoặc có xung đột giữa các hàm phân loại khác nhau. Trong phân loại đa nhãn cho đồ thị mỗi đồ thị con thường xuyên đóng thể hiện một tập hợp các nhãn mà có thể gán cho đồ thị nên có thể một tập đồ thị con sẽ ứng với một tập nhãn do đó có sự chồng đè các nhãn lên nhau trong quá trình phân loại. Vì vậy, nghiên cứu sinh lựa chọn áp dụng lý thuyết Dempster-Shafer cho phân loại đa nhãn đồ thị. Công trình [24] đề xuất mô hình phân loại đa nhãn cho các đối tượng theo một độ đo tương tự giữa các đối tượng theo lý thuyết Dempster-Shafer. Nghiên cứu sinh kết hợp mô hình phân loại đa nhãn trong [24] với độ đo tương tự trên dàn giao khái niệm cho cơ sở dữ liệu đồ thị giao tác để thực hiện phân loại đa nhãn cho đồ thị đạt được độ chính xác và hiệu năng cao.

### 3.3.1 Ý tưởng đề xuất

Phân loại đa nhãn dựa trên lý thuyết Dempster Shafer được thực hiện trên dữ liệu dạng véctơ. Mô hình phân loại đa nhãn dựa trên véctơ khi áp dụng cho đồ thị gặp khó

khăn do đồ thị thiếu biểu diễn véctơ. Dựa trên biểu diễn véctơ, các mô hình phân loại đa nhãn đưa ra các độ đo tương tự với mục đích tìm được tập k-láng giềng gần nhất. Sau khi tìm được tập k-láng giềng gần nhất sẽ thực hiện phân loại cho một đối tượng mới tập nhãn dựa trên sự kết hợp các hàm khối các tập nhãn theo luật Dempster của tập k-láng giềng gần nhất. Khi áp dụng cho cơ sở dữ liệu đồ thị, việc xây dựng các hàm khối để xác định tập k-láng giềng gần nhất dựa trên véctơ là khó có thể thực hiện được. Nghiên cứu sinh đã đề xuất việc xây dựng một dàn giao khái niệm mà trên đó có một độ đo có thể xây dựng được tập hàm khối k-láng giềng gần nhất. Công việc này giúp cho việc phân loại đa nhãn đồ thị có thể được thực hiện dễ dàng. Phần này trình bày một số thủ tục, hàm, thuật toán cho việc xây dựng dàn giao khái niệm và áp dụng cho phân loại đa nhãn đồ thị.

- Dựa trên thuật toán khai phá đồ thị con thường xuyên đóng PSI-CFSM, thủ tục *CalculateFCT* thực hiện việc xây dựng bảng ngữ cảnh chính thức từ tập các đồ thị con thường xuyên đóng.
- Thủ tục *CalculateIcebergeLattice* xây dựng dàn giao khái niệm từ bảng ngữ cảnh chính thức được sinh bởi *CalculateFCT*
- Định nghĩa đường đi giữa hai đồ thị $g_i, g_j$ trên dàn giao khái niệm IcebergLattice.
- Định nghĩa độ đo trên dàn giao khái niệm IcebergLattice.
- Thuật toán *DSMLGC* [3] dùng để xác định tập nhãn theo luật Dempster cho một đối tượng đồ thị $g_x$ từ các tập nhãn hàm khối của tập k-láng giềng gần nhất theo độ đo tương tự trên dàn giao khái niệm IcebergLattice.

### 3.3.2 Xây dựng dàn giao khái niệm

Để xây dựng dàn giao khái niệm trên đồ thị dựa trên tập đối tượng là tập các đồ thị trong cơ sở dữ liệu đồ thị giao tác và tập thuộc tính là tập đồ thị con thường xuyên của các đồ thị trong cơ sở dữ liệu đồ thị giao tác. Với một cơ sở dữ liệu đồ thị sẽ có một số lượng hàm mũ các đồ thị con thường xuyên do vậy tìm tất cả các khái niệm chính thức sẽ có độ phức tạp tính toán thời gian hàm mũ. Để hạn chế các khái niệm chính thức, nghiên cứu sinh giới hạn tập thuộc tính theo tập các đồ thị con thường xuyên đóng theo thuật toán PSI-CFSM. Dựa trên mối quan hệ khái niệm cha và khái niệm con từ ngữ cảnh chính thức gồm tập thuộc tính là các đồ thị con thường xuyên đóng, xây dựng dàn giao khái niệm là một đồ thị vô hướng có các cạnh độ dài bằng một đơn vị với các đỉnh là các khái niệm chính thức và các cạnh là mối quan hệ cha con giữa các khái niệm chính thức.

Cho một cơ sở dữ liệu đồ thị $\mathbb{GD}$, để xây dựng dàn giao cho các đồ thị sử dụng một *bảng ngữ cảnh chính thức* theo định nghĩa 1.5.1 bằng cách xây dựng tập tất cả các đồ thị con thường xuyên đóng $CS$ của cơ sở dữ liệu đồ thị $\mathbb{GD}$ và coi tập $CS$ là tập các thuộc tính còn cơ sở dữ liệu đồ thị tập đối tượng. Mối quan hệ giữa tập đối tượng và tập thuộc tính thể hiện qua việc một đồ thị $G_i \in \mathbb{GD}$ có chứa một đồ thị con thường xuyên đóng $g_j \in CS$ thì đồ thị $G_i$ và đồ thị con thường xuyên đóng $g_j$ là có mối quan hệ với nhau.

---

**Procedure** CalculateFCT($CS$)

---

**Đầu vào:** $\mathbb{GD}$

**Đầu ra :** Bảng ngữ cảnh chính thức $FCT$

1 Xây dựng $CS_i$ là các đồ thị con thường xuyên đóng của $g_i \in \mathbb{GD}$ theo PSI-CFSM;

2 Từ các $CS_i$ xây dựng bảng $FCT$;

3 Trả về $FCT$;

---

Từ bảng ngữ cảnh chính thức, tìm ra tất cả các *khái niệm chính thức* theo định nghĩa 1.5.2. Theo định nghĩa 1.5.3, xây dựng được một dàn giao khái niệm IcebergLattice.

**Procedure** CalculateIcebergLattice($FCT$)

**Đầu vào:** Bảng ngữ cảnh chính thức $FCT$

**Đầu ra :** Dàn giao khái niệm IcebergLattice

1 Từ $FCT$ xây dựng các khái niệm chính thức $KCT_i$ và quan hệ giữa các khái niệm chính thức $QCT_j$;
2 **foreach** $KCT_i$ **do**
3 | Tạo dựng các nút $KCT_i$ trên dàn giao khái niệm;
4 **end**
5 **foreach** $QCT_j$ **do**
6 | Tạo các liên kết giữa các $KCT_i$ nếu $KCT_i \in QCT_j$;
7 **end**
8 Trả về $IcebergLattice$;

Dựa trên định nghĩa 1.5.3, 1.4.2 xây dựng dàn giao khái niệm Iceberglattice $CL$ sẽ luôn có phần tử *cận trên nhỏ nhất* và *cận dưới lớn nhất* cho mỗi cặp phần tử trên dàn giao khái niệm. Từ dàn giao khái niệm, định nghĩa một độ đo dựa trên khoảng cách tính theo số lượng cạnh tính từ phần từ nhỏ nhất cận dưới $lub(x, y)$ đến mỗi đỉnh $x, y \in CL$ trên dàn giao khái niệm gọi là $d(x, y)$.

**Định nghĩa 3.3.1.** [3] Đường đi giữa hai đỉnh $x, y$ trên dàn giao khái niệm $CL$ là tổng các đường đi ngắn nhất từ $lub(x, y)$ đến $x$ và từ $lub(x, y)$ đến $y$.

**Bổ đề 3.3.2.** [3] Đường đi giữa hai đỉnh $x, y$ theo định nghĩa 3.3.1 là đường đi ngắn nhất.

*Chứng minh.* Gọi đường đi ngắn nhất giữa hai đỉnh $x, y$ trên dàn giao khái niệm $CL$ là $shortest\_path(x, y)$. Giả sử rằng $shortest\_path(x, y)$ là đường đi không đi

qua $lub(x, y$. Như vậy, trên đường đi từ $x$ đến $y$ sẽ là một đường đi trực tiếp. Theo định nghĩa dàn giao khái niệm 1.5.3, thì giữa cặp đỉnh bất kỳ $(x, y)$ trên dàn giao nếu có đường đi thì phải có quan hệ khái niệm con - khái niệm cha. Nghĩa là tồn tại một $lub(x, y)$ hay nói cách khác đường đi trực tiếp giữa cặp $(x, y)$ chỉ xảy ra khi $lub(x, y) = x$ hoặc $lub(x, y) = y$. Như vậy sẽ mâu thuẫn với giả thiết là đường đi giữa cặp $(x, y)$ không đi qua $lub(x, y)$. Vậy giả thiết là sai. Bổ đề được chứng minh. □

**Định nghĩa 3.3.3.** [3] Cho $CL$ là một dàn giao khái niệm, độ đo tương tự giữa hai đồ thị $g_i, g_j \in \mathbb{GD}$ là đường đi giữa hai đỉnh khái niệm chính thức chứa $g_i, g_j$.

$$d(g_i, g_j) = |shortest\_path(c(g_i), c(g_j))|$$

với $c(g_i), c(g_j)$ là các khái niệm chính thức của các đồ thị $g_i, g_j$ trong ngữ cảnh chính thức của cơ sở dữ liệu đồ thị $\mathbb{GD}$.

**Định lý 3.3.4.** [3] $d(g_i, g_j)$ thỏa mãn tính chất của độ đo tương tự theo khoảng cách.

*Chứng minh.* Để chứng minh $d(g_i, g_j)$ thỏa mãn là một độ đo tương tự, cần phải chứng minh các tính chất sau:

$$1) d(g_i, g_i) = 0, \forall g_i \in CL$$
$$2) d(g_i, g_j) = d(g_j, g_i), \forall g_i, g_j \in CL$$
$$3) d(g_i, g_j) \leqslant d(g_i, g_k) + d(g_k, g_j), \forall g_i, g_j, g_k \in CL$$

Tính chất 1) là rõ ràng vì trên dàn giao khái niệm của $\mathbb{GD}$ chỉ có duy nhất một khái niệm chính thức cho $g_i$ nên $lub(g_i, g_i) = g_i$ và $d(g_i, g_i) = 0$

Tính chất 2) là rõ ràng vì với cặp đỉnh $(g_i, g_j)$ trên dàn giao khái niệm chỉ chứa duy nhất một khái niệm chính thức $lub(g_i, g_j)$ nên $lub(g_i, g_j) = lub(g_j, g_i)$ do đó dẫn đến $|shortest\_path(c(g_i), c(g_j))| = |shortest\_path(c(g_j), c(g_i))|$ và $d(g_i, g_j) = d(g_j, g_i)$.

Bây giờ cần chứng minh tính chất 3): Bằng phương pháp phản chứng. Giả sử rằng $d(g_i, g_j) > d(g_i, g_k) + d(g_k, g_j)$. Theo định nghĩa 3.3.1 và bổ đề 3.3.2 thì $d(g_i, g_k) = |shortest\_path(c(g_i), c(g_k))|$ và $d(g_k, g_j) = |shortest\_path(c(g_k), c(g_j))|$. Tuy nhiên, $d(g_i, g_j) = |shortest\_path(c(g_i), c(g_j))|$ nên $d(g_i, g_k) + d(g_k, g_j)$ không thể nhỏ hơn $d(g_i, g_j)$. Điều này trái với giả thiết ban đầu. Định lý được chứng minh. □

### 3.3.3 Thuật toán phân loại đa nhãn đồ thị

Theo Grabisch [36], hàm niềm tin $bel$ trên poset $(L, \leqslant)$ là hàm $f$ trong biến đổi Mobius theo (1.7) nếu $bel(\perp) = 0$ và $bel(\top) = 1$, hàm khối $m$ tìm được qua hàm $f$ theo (1.8), hàm niềm tin $bel$ là đơn điệu theo (1.15) và hàm tính chất chung $q$ là đồng biến đổi Mobius của $bel$ theo (1.14). [36] hầu hết các kết quả của lý thuyết Dempster-Shafer đều có thể áp dụng lên các dàn giao (theo định nghĩa 1.4.2) nói chung.

Theo định nghĩa 1.4.2 về tập có thứ tự (poset) và dàn giao, cho tập dữ liệu đồ thị $\mathbb{GD}$ và tập nhãn gắn với mỗi đồ thị, có thể coi tập tất cả tập nhãn là $2^L$ với tập nhãn $L = \{l_1, l_2, ..., l_Q\}$, $|L| = Q$. Quan hệ $A \leqslant B \Leftrightarrow A \subseteq B$ trên $\Omega = 2^L$ với mọi $A, B \subseteq 2^L$ có phần tử nhỏ nhất là $\perp = \varnothing$ và phần tử lớn nhất là $\top = L$ là một dàn giao (gọi là dàn giao $(2^L, \subseteq)$) với toán tử $\wedge = \cap$ và $\vee = \cup$. Khi tập $L$ rất lớn, khả năng làm việc với dàn giao $(2^L, \subseteq)$ là không khả thi.

Cho một khung phân biệt $\Omega$. Một tập con $I$ của $\Omega$ là một dàn giao khoảng giới hạn nếu tồn tại các phần từ $a$ và $b$ của $\Omega$ mà $I = \{x \in \Omega | a \leqslant x \leqslant b\}$. Ký hiệu $I$ là $[a, b]$. Rõ ràng $\Omega$ là khoảng $[\perp, \top]$ và $\varnothing$ là khoảng rỗng. Cho $\Theta_{(\Omega, \leqslant)} \subseteq 2^\Omega$ gọi là tập các khoảng bao gồm cả tập rỗng $\varnothing$

$$\Theta_{(\Omega, \leqslant)} = \{[a, b] | a, b \in \Omega, a \leqslant b\} \cup \{\varnothing\}$$

Giao của hai khoảng là một khoảng

$$[a,b] \cap [c,d] = \begin{cases} [a \vee c, b \wedge d], & if\ a \vee c \leqslant b \wedge d \\ \varnothing & otherwise. \end{cases} \tag{3.10}$$

Theo định nghĩa 1.4.5 thì $\Theta_{(\Omega,\leqslant)}$ là một hệ đóng và $(\Theta_{(\Omega,\leqslant)}, \subseteq)$ là một dàn giao với phần tử nhỏ nhất là $\varnothing$, phần tử lớn nhất là $\Omega$ và $([a,b] \subseteq [c,d] \Leftrightarrow c \leqslant a \wedge b \leqslant d)$. Toán tử *meet* là phép giao (3.10) và toán tử *join* là phép $\sqcup$ được xác định như sau.

$$[a,b] \sqcup [b,d] = [a \wedge c, b \vee d]. \tag{3.11}$$

Xét cấu trúc dàn giao với $\Omega = 2^L$ với thứ tự $\subseteq$. Các hàm niềm tin trên dàn giao $(2^{2^L}, \subseteq)$ là không thể thực hiện khi có lượng phần tử là hai lần hàm mũ. Như vậy, dùng khái niệm khoảng $A \subseteq B$ với hai tập con $A, B$ của $2^L$ là khoảng $[A, B]$

$$[A,B] = \{C | C \in 2^L, A \subseteq C \subseteq B\}.$$

Tập hợp các khoảng trong dàn giao $(\Omega, \subseteq)$ là $\Theta_{(\Omega,\leqslant)} = \{[A,B] | A, B \subseteq \Omega, A \subseteq B\} \cup \varnothing_\Omega$ với $\varnothing_\Omega$ là các tập rỗng của $\Omega$ khác với các tập rỗng của $L$. Rõ ràng $\Theta_{(\Omega,\subseteq)} \subseteq 2^\Omega = 2^{2^L}$. Khoảng $[A, B]$ có thể được xem như chắc chắn chứa các nhãn nằm trong $A$ và có thể chứa các nhãn nằm trong $B$ và chắc chắn không chứa các nhãn nằm trong $\bar{B}$. Toán tử *meet* và *join* của $L, \subseteq$ là các phép giao và hợp tương ứng sẽ có các toán tử cho $(\Theta_{(\Omega,\subseteq)}, \subseteq)$ dựa trên các công thức (3.10) và (3.11) như sau.

$$[A,B] \cap [C,D] = \begin{cases} [A \cup C, B \cap D], & if\ A \cup C \subseteq B \cap D \\ \varnothing_\Omega & otherwise. \end{cases} \tag{3.12}$$

$$[A,B] \sqcup [C,D] = [A \cap C, B \cup D]. \tag{3.13}$$

Thuật toán phân loại đa nhãn cho đồ thị [3] được xây dựng theo phương pháp k-láng giềng gần nhất để xác định tập nhãn cho đồ thị $g_n \in \mathbb{GD}$ chưa có nhãn với mọi đồ thị $G_i \in \mathbb{GD}$ đã được gán nhãn $L_i \subseteq L$. Tương ứng với mỗi đồ thị $g_i \in kNN(g_n)$ sẽ là một hàm niềm tin với khoảng nhãn $[A_i, B_i]$ được xác định theo dàn giao khái niệm $CL$ với $A_i$ là tập nhãn của $g_i$ và $B_i$ là tập nhãn của $lub(g_i, g_n)$. Luật bằng chứng k láng giềng gần nhất [23], cho $\phi_x$ là tập k láng giềng gần nhất của mẫu đối tượng mới được mô tả bởi véctơ đặc trưng $x$ theo một độ đo tương tự $d$ và $x_i$ là một phần tử của tập k láng giềng gần nhất có tập nhãn nằm trong khoảng $[A_i, B_i]$ (poset hữu hạn cục bộ) thì một mục bằng chứng có thể được mô tả như hàm khối sau:

$$m_i([A_i, B_i]) = \alpha_i, \tag{3.14}$$

$$m_i([\varnothing_\Gamma, \Gamma]) = 1 - \alpha_i \tag{3.15}$$

với $\alpha_i$ là độ đo tương tự dựa trên công thức (3.3.3) theo tỉ lệ đối với tổng khoảng cách tất cả các đồ thị $g_k$ tới $g_n$.

Theo [25], [97] đề xuất luật để xác định tập nhãn cho đối tượng $x$. Cho $\hat{Y}$ là tập nhãn dự đoán sẽ được gán cho $x$. Để quyết định mỗi nhãn $\theta \in \Gamma$ được gán cho $x$ hay không, hai số lượng được tính là cấp độ hàm niềm tin $bel([\{\theta\}, \Gamma])$, $\hat{Y}$ là tập nhãn

**Algorithm 8:** DSMLGC($DS$)

**Đầu vào:** $\mathbb{GD}$, $L$, $g_x$

**Đầu ra :** $A \subseteq L$ là tập nhãn của $g_x$

1 Xây dựng dàn giao khái niệm $IcebergLattice$ cho $\mathbb{GD}$ và $g_x$;

2 Xác định k-láng giềng của $g_x$ trên $IcebergLattice$ là tập $kNN(g_x)$;

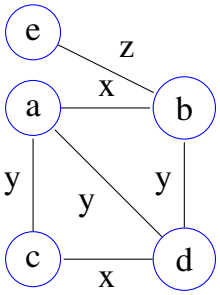
3 Áp dụng luật Dempster-Shafer tìm tập nhãn cho $g_x$ từ $kNN(g_x)$;

đúng chứa $\theta$, và cấp độ hàm niềm tin $bel([\varnothing, \{\bar{\theta}\}])$ mà không chứa $\theta$. Tập nhãn dự đoán được gán $\hat{Y}$ được xác định như sau:

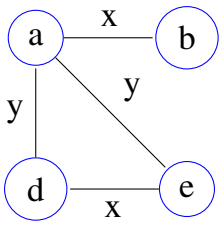
$$\hat{Y} = \{\theta \in \Gamma \mid bel([\{\theta\}, \Gamma]) \geqslant bel([\varnothing, \{\bar{\theta}\}])\}. \tag{3.16}$$

## 3.4 Ví dụ PSI-CFSM và phân loại đa nhãn

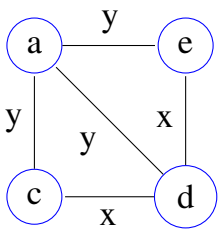
Cho cơ sở dữ liệu đồ thị $\mathbb{GD} = \{g_1, g_2, g_3, g_4\}$ với tập nhãn $L = \{l_1, l_2, l_3, l_4, l_5\}$, ngưỡng độ hỗ trợ $\sigma = 50\%$:



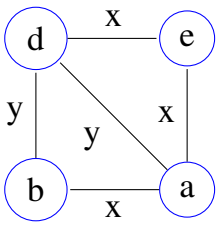

$g_1, \{l_1, l_2\}$




$g_2, \{l_2, l_4\}$

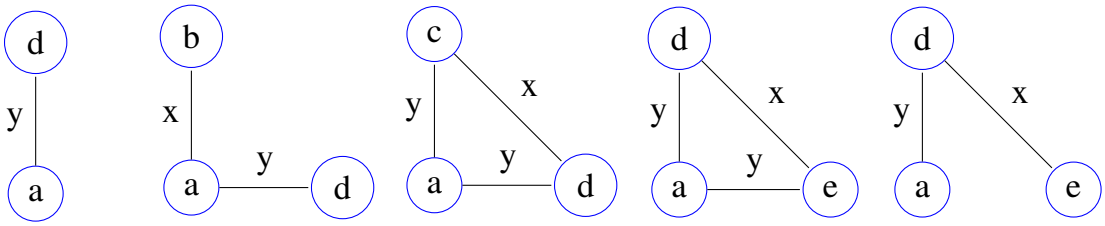



$g_3, \{l_3\}$




$g_4$

Cơ sở dữ liệu đồ thị $\mathbb{GD}$ có $|\mathbb{GD}| = 4$, với ngưỡng độ hỗ trợ $\sigma = \frac{2}{4} = 50\%$, tất cả những đồ thị con bất kỳ $sg$ có độ hỗ trợ $sup_{sg} >= \frac{2}{4}$ đều là các đồ thị con thường xuyên. Áp dụng thuật toán PSI-CFSM tìm được tập tất cả các đồ thị con thường xuyên đóng $CS = \bigcup_{i=2}^{k} \{sg \in CS_k\}$ của $\mathbb{GD}$.




$\left[\begin{smallmatrix} a \\ y & d \end{smallmatrix}\right] \frac{4}{4}, 15$ $\left[\begin{smallmatrix} a & & b \\ x & & \\ y & 0 & d \end{smallmatrix}\right] \frac{2}{4}, 33$ $\left[\begin{smallmatrix} a & & c \\ y & & \\ y & x & d \end{smallmatrix}\right] \frac{2}{4}, 45$ $\left[\begin{smallmatrix} a & & d \\ y & & \\ y & x & e \end{smallmatrix}\right] \frac{2}{4}, 41$ $\left[\begin{smallmatrix} a & & d \\ y & & \\ 0 & x & e \end{smallmatrix}\right] \frac{3}{4}, 27$

Ngữ cảnh chính thức là tập tất cả mối quan hệ giữa mọi đồ thị $g_i \in \mathbb{GD}$ với tập tất cả đồ thị con thường xuyên đóng $CS$:

Bảng 3.1: Quan hệ giữa đồ thị và tập tất cả đồ thị con thường xuyên đóng

| OA | $\left[\begin{smallmatrix} a \\ y & d \end{smallmatrix}\right]$ | $\left[\begin{smallmatrix} a & & b \\ x & & \\ y & 0 & d \end{smallmatrix}\right]$ | $\left[\begin{smallmatrix} a & & c \\ y & & \\ y & x & d \end{smallmatrix}\right]$ | $\left[\begin{smallmatrix} a & & d \\ y & & \\ y & x & e \end{smallmatrix}\right]$ | $\left[\begin{smallmatrix} a & & d \\ y & & \\ 0 & x & e \end{smallmatrix}\right]$ |
|---|---|---|---|---|---|
| $g_1$ | x | x | x | | |
| $g_2$ | x | x | | x | x |
| $g_3$ | x | | x | x | x |
| $g_4$ | x | x | | | x |

Khái niệm chính thức của ngữ cảnh chính thức:

$$\left(\{g_1\}, \left\{\left[\begin{smallmatrix} a \\ y & d \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & b \\ x & & \\ y & 0 & d \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & c \\ y & & \\ y & x & d \end{smallmatrix}\right]\right\}\right)$$

$$\left(\{g_2\}, \left\{\left[\begin{smallmatrix} a \\ y & d \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & b \\ x & & \\ y & 0 & d \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & d \\ y & & \\ y & x & e \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & d \\ y & & \\ 0 & x & e \end{smallmatrix}\right]\right\}\right)$$

$$\left(\{g_3\}, \left\{\left[\begin{smallmatrix} a \\ y & d \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & c \\ y & & \\ y & x & d \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & d \\ y & & \\ y & x & e \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & d \\ y & & \\ 0 & x & e \end{smallmatrix}\right]\right\}\right)$$

$$\left(\{g_1, g_3\}, \left\{\left[\begin{smallmatrix} a \\ y & d \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & c \\ y & & \\ y & x & d \end{smallmatrix}\right]\right\}\right)$$

$$\left(\{g_2, g_3\}, \left\{\left[\begin{smallmatrix} a \\ y & d \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & d \\ y & & \\ y & x & e \end{smallmatrix}\right], \left[\begin{smallmatrix} a & & d \\ y & & \\ 0 & x & e \end{smallmatrix}\right]\right\}\right)$$

$$\left(\{g_2, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right)$$

$$\left(\{g_1, g_2, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}\right\}\right)$$

$$\left(\{g_2, g_3, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right)$$

$$\left(\{g_1, g_2, g_3, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}\right\}\right)$$

$$\left(\{\varnothing\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right)$$

Mối quan hệ giữa các khái niệm chính thức

$$\left(\{g_1\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}\right\}\right) \subseteq \left(\{g_1, g_3\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}\right\}\right)$$

$$\left(\{g_3\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_1, g_3\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}\right\}\right)$$

$$\left(\{g_2\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_2, g_3\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right)$$

$$\left(\{g_3\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_2, g_3\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right)$$

$$\left(\{g_2\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_2, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right)$$

$$\left(\{g_2, g_3\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_2, g_3, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right)$$

$$\left(\{g_2, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_2, g_3, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right)$$

$$\left(\{g_1\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}\right\}\right) \subseteq \left(\{g_1, g_2, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}\right\}\right)$$

$$\left(\{g_2, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_1, g_2, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}\right\}\right)$$

$$\left(\{g_1, g_3\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}\right\}\right) \subseteq \left(\{g_1, g_2, g_3, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}\right\}\right)$$

$$\left(\{g_1, g_2, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}\right\}\right) \subseteq \left(\{g_1, g_2, g_3, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}\right\}\right)$$

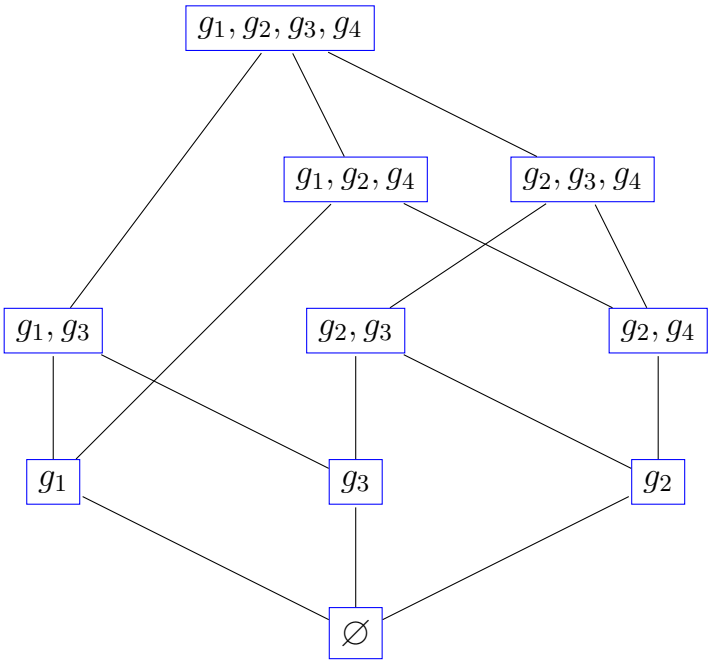
$$\left(\{g_2, g_3, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_1, g_2, g_3, g_4\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}\right\}\right)$$

$$\left(\{\varnothing\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_1\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}\right\}\right)$$

$$\left(\{\varnothing\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_2\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right)$$

$$\left(\{\varnothing\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ x & b \\ y & 0 & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right) \subseteq \left(\{g_3\}, \left\{\begin{bmatrix} a \\ y & d \end{bmatrix}, \begin{bmatrix} a \\ y & c \\ y & x & d \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ y & x & e \end{bmatrix}, \begin{bmatrix} a \\ y & d \\ 0 & x & e \end{bmatrix}\right\}\right)$$

Từ mối quan hệ cha con giữa các khái niệm chính thức ta xây dựng một dàn giao khái niệm:




Hình 3.4: Dàn giao khái niệm $CL$ của các đồ thị $g_i \in \mathbb{GD}$

Để xác định nhãn cho đồ thị $g_4 \in \mathbb{GD}$, dựa vào dàn giao khái niệm $CL$ tìm k-láng giềng gần nhất của $g_4$ với $k = 2$ trên tập dữ liệu đồ thị $\mathbb{GD}$ là các đồ thị $g_1, g_2, g_3$ theo đường đi ngắn nhất từ đỉnh chung đến hai đồ thị cần so sánh.

$d(g_4, g_1) = \frac{2}{6} = 0.3333$, $d(g_4, g_3) = \frac{3}{6} = 0.5000$, $d(g_4, g_2) = \frac{1}{6} = 0.1667$

Như vậy, xác định được k-láng giềng gần nhất với $g_4$ với $k = 2$ tương ứng là $g_1, g_2$.

Xác định khoảng nhãn tương ứng với k-láng giềng gần nhất $g_1, g_2$ là $[\{l_1, l_2\}, \{l_1, l_2, l_4\}]$, $[\{l_2, l_4\}, \{l_2, l_4\}]$

Tính được các hàm khối như sau:

$m_{g_1}([\{l_1, l_2\}, \{l_1, l_2, l_4\}]) = 0.33$

$m_{g_1}([\varnothing, \mathbb{L}]) = 1 - 0.33 = 0.67$

$m_{g_2}([\{l_2, l_4\}, \{l_2, l_4\}]) = 0.17$

$m_{g_2}([\varnothing, \mathbb{L}]) = 1 - 0.17 = 0.83$

Sử dụng luật Dempster thu được kết quả sau:

Bảng 3.2: Luật Dempster kết hợp các hàm cấp phát khối

| m | $m_{g_1}([\{l_1, l_2\}, \{l_1, l_2, l_4\}])$ | $m_{g_1}([\varnothing, \mathbb{L}])$ |
|---|---|---|
| $m_{g_2}([\{l_2, l_4\}, \{l_2, l_4\}])$ | $m([\{l_1, l_2, l_4\}, \{l_1, l_2, l_4\}])$ | $m([\{l_2, l_4\}, \{l_2, l_4\}])$ |
| $m_{g_2}([\varnothing, \mathbb{L}])$ | $m([\{l_1, l_2\}, \{l_1, l_2, l_4\}])$ | $m([\varnothing, \mathbb{L}])$ |

Dựa vào luật Dempster, tính được:

$m([\{l_1, l_2, l_4\}, \{l_1, l_2, l_4\}]) = 0.06$

$m([\{l_2, l_4\}, \{l_2, l_4\}]) = 0.11$

$m([\{l_1, l_2\}, \{l_1, l_2, l_4\}]) = 0.27$

$m([\varnothing, \mathbb{L}]) = 0.56$

$bel([\{l_1\}, \mathbb{L}]) = 0.06 + 0.27 = 0.33 > bel([\varnothing, \{\bar{l_1}\}]) = 0.11$, $g_4$ có nhãn $l_1$

$bel([\{l_2\}, \mathbb{L}]) = 0.33 > bel([\varnothing, \{\bar{l_2}\}]) = 0$, $g_4$ có nhãn $l_2$

$bel([\{l_3\}, \mathbb{L}]) = 0 < bel([\varnothing, \{\bar{l_3}\}]) = 0.11$

$bel([\{l_4\}, \mathbb{L}]) = 0.06 > bel([\varnothing, \{\bar{l_4}\}]) = 0$, $g_4$ có nhãn $l_4$

$bel([\{l_5\}, \mathbb{L}]) = 0 < bel([\varnothing, \{\bar{l_5}\}]) = 0.11$

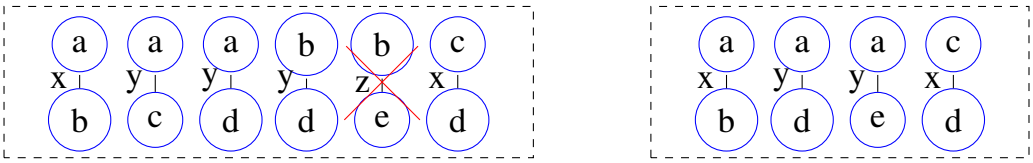
Như vậy tập nhãn của đồ thị $g_4$ được xác định là $\{l_1, l_2, l_4\}$

## 3.5 Đánh giá thử nghiệm

Phần này trình bày so sánh kết quả thực nghiệm giữa thuật toán PSI-CFSM và gSpan trên một số bộ dữ liệu. Thực nghiệm chứng tỏ rằng phương pháp khai phá đồ thị con thường xuyên đóng PSI-CFSM của nghiên cứu sinh đề xuất tối ưu về mặt thời gian tính toán hơn gSpan nhờ vấn đề xác định đẳng cấu đồ thị con trong thời gian đa thức. Sử dụng các bộ dữ liệu Chemical Compound đi kèm với thuật toán gSpan và bộ dữ liệu nghiên cứu sinh tự sinh trong phần ví dụ, cùng với việc đặt các ngưỡng độ hỗ trợ tối thiểu khác nhau để so sánh thời gian thực hiện 2 thuật toán PSI-CFSM và gSpan. Kết quả được cho trong bảng 3.3.

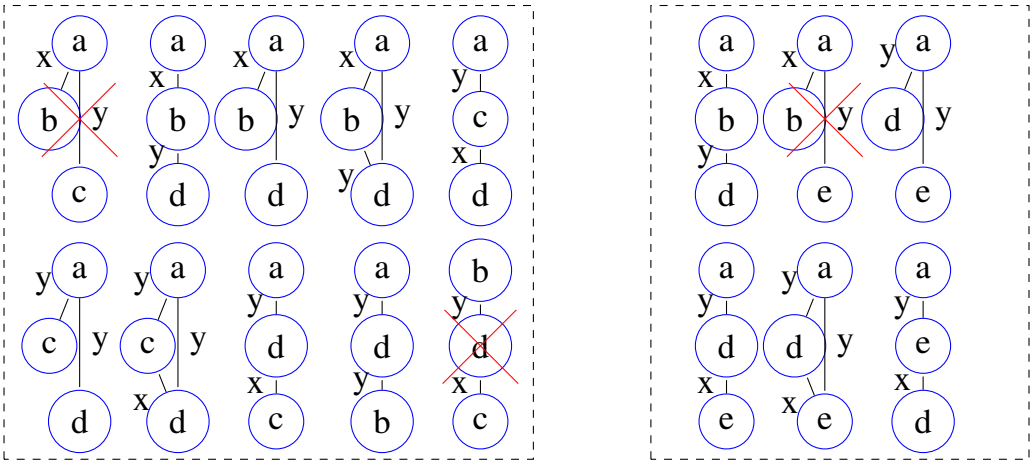
Phân tích rõ hơn về tốc độ thực hiện của thuật toán PSICFSM nhanh hơn thuật toán gSpan

- Thuật toán PSI-CFSM sinh ra tất cả các đồ thị con của đồ thị $g_i \in GD$ và tỉa khi không thoả mãn độ đo thường xuyên.




$(C^2_{g_1})$ $(C^2_{g_2})$

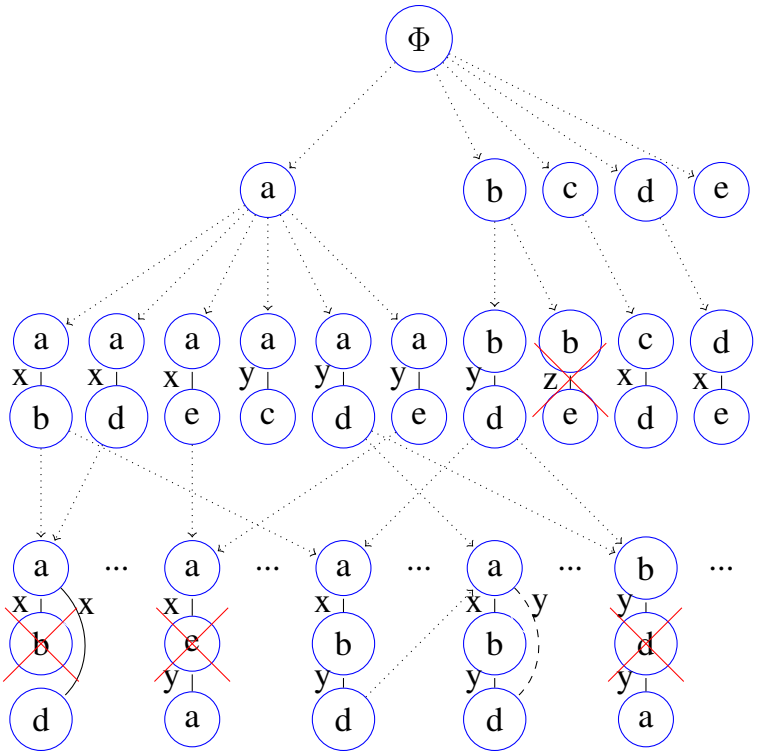
Hình 3.5: Sinh ứng viên và tỉa đồ thị con 2-subgraph theo PSI-CFSM



$(C^3_{g_1})$ $(C^3_{g_2})$

Hình 3.6: Sinh ứng viên và tỉa đồ thị con 3-subgraph theo PSI-CFSM

- Thuật toán gSpan dựa trên cây DFS Code Tree để sinh ra một đồ thị con ứng viên và thực hiện các bước tỉa khi không thoả mãn DFSC hoặc không thoả mãn độ đo thường xuyên.




Hình 3.7: Tỉa các đồ thị con ứng viên: không thường xuyên, không thoả mãn DFSC

Có thể dễ dàng nhận ra ngay rằng có rất nhiều đồ thị con ứng viên được sinh ra trên cây DFS Code Tree mà không thuộc tập đồ thị con của $g_i \in GD$, chẳng hạn như đồ thị $axe0ya \not\subset g_i, \forall g_i \in GD$. Chính điều này là nguyên nhân dẫn đến gSpan kém hiệu quả so với PSICFSM khi tập dữ liệu đồ thị càng ngày càng lớn. Bảng 3.3 kết quả thực nghiệm chỉ rõ điều này.

Bộ dữ liệu được chọn trong thực nghiệm là bộ dữ liệu Chemical Compound được chính thuật toán gSpan lựa chọn sử dụng đi kèm được public trên mạng internet. Bộ dữ liệu này phù hợp với khai phai phá dữ liệu đồ thị để thể hiện sự phức tạp khi biểu diễn đồ thị và điển hình cho phân tích đẳng cấu đồ thị con. Rất nhiều các công trình công bố chọn lựa bộ dữ liệu Chemical Compound cho khai phá dữ liệu đồ thị.

Bảng 3.3: Khai phá đồ thị con thường xuyên (đơn vị thời gian: giây)

| Ngưỡng (số lần xuất hiện) | Thuật toán | Dữ liệu 4 đồ thị | Dữ liệu 50 đồ thị |
|---|---|---|---|
| 2 | gSpan | 0.07s | 1120s |
| 2 | PSI-CFSM | 0.027s | 66.2s |
| 5 | gSpan | 0.0s | 3.26s |
| 5 | PSI-CFSM | 0.006s | 2.986s |
| 10 | gSpan | 0.0s | 1.74s |
| 10 | PSI-CFSM | 0.006s | 1.42s |

## 3.6 Kết luận chương

Đẳng cấu đồ thị con trong khai phá đồ thị con thường xuyên là một trong những vấn đề có độ phức tạp thời gian không đa thức. Dựa trên một số điều kiện ràng buộc, nghiên cứu sinh đã đề xuất một phương pháp xác định đẳng cấu đồ thị con trong thời gian đa thức. Do đó vấn đề khai phá đồ thị con thường xuyên được tối ưu về mặt thời gian tính toán. Tiếp nối sự cải thiện về mặt thời gian trong khai phá đồ thị con thường xuyên đóng, nghiên cứu sinh đề xuất độ đo tương tự giữa các đồ thị trên dàn giao khái niệm dựa trên kết quả khai phá đồ thị con thường xuyên đóng và dùng độ đo được đề xuất kết hợp với mô hình phân loại đa nhãn dựa trên lý thuyết Dempster Shafer để thực hiện phân loại đồ thị đa nhãn chính xác và hiệu quả. Ngoài việc chứng minh tính đúng đắn và đầy đủ các nghiên cứu khoa học của nghiên cứu sinh về khai phá đồ thị con thường xuyên đóng và phân loại đa nhãn đồ thị, nghiên cứu sinh cũng chứng tỏ được tính hiệu quả của phương pháp đề xuất bằng một số thực nghiệm đã được kiểm chứng.

# KẾT LUẬN, KIẾN NGHỊ

**Kết quả chính của luận án**

Dữ liệu ngày càng tăng nhanh trong các tổ chức, công ty, chính phủ dẫn đến một nhu cầu khai phá dữ liệu lớn ở tốc độ cao [101]. Khai phá dữ liệu lớn thúc đẩy phát triển ứng dụng ở nhiều lĩnh vực quan trọng như: y tế, giao thông, tài chính, giáo dục, [28], [63] nhằm đem lại lợi ích trong việc hỗ trợ ra quyết định, cắt giảm chi phí, tăng lợi nhuận hoặc tạo ra các sản phẩm, dịch vụ mới. Khai phá dữ liệu lớn gặp nhiều khó khăn do đó luận án tập trung nghiên cứu, phát triển một số phương pháp khai phá dữ liệu trên dữ liệu có cấu trúc cụ thể là cấu trúc dạng bảng và dạng đồ thị và đã đạt được một số kết quả về rút gọn thông tin và tối ưu tính toán.

Dữ liệu lớn dẫn đến nhu cầu rút gọn dữ liệu để giảm không gian lưu trữ và tối ưu thời gian tính toán. Các công trình nghiên cứu tập trung vào tìm các rút gọn thuộc tính theo lý thuyết tập thô của Pawlak [64] trên bảng quyết định. Tìm tất cả các rút gọn thuộc tính có độ phức tạp thời gian hàm mũ [48] $O(m * 2^n)$ với $m$ là số lượng đối tượng và $n$ là số lượng thuộc tính của bảng quyết định nhất quán. Luận án phát hiện phương pháp tìm một rút gọn đối tượng $m' < m$ trong thời gian đa thức mà vấn đề tìm tất cả các rút gọn thuộc tính được bảo toàn. Theo đó, độ phức tạp tính toán tìm tất cả các rút gọn thuộc tính chỉ còn là $O(m' * 2^n)$ và giảm không gian lưu trữ dữ liệu đặc biệt đối với dữ liệu lớn. Ngoài ra, để giảm độ phức tạp tính toán hàm mũ trong vấn đề sinh luật quyết định, sinh cây quyết định thì các nghiên cứu công bố tìm một rút gọn thuộc tính heuristic trong thời gian đa thức. Thêm vào đó luận án thành công tìm một rút gọn thuộc tính trong thời gian đa thức không heuristic và một phương pháp cải tiến sinh cây quyết định có tốc độ thực hiện nhanh hơn thuật toán sinh cây quyết định ID3 trên bảng quyết định nhất quán. Trong luận án, các đề xuất của nghiên cứu sinh được chứng minh đúng đắn và đầy đủ cùng với thực nghiệm chứng tỏ thuật toán sinh cây quyết định của nghiên cứu sinh nhanh hơn thuật toán ID3.

Dữ liệu lớn là dữ liệu được thu thập từ nhiều miền, nhiều lĩnh vực do đó có đa dạng cấu trúc biểu diễn khác nhau. Các thuật toán khai phá dữ liệu chỉ có thể khai phá dữ liệu trên một tập dữ liệu thống nhất về kiểu dạng biểu diễn. Các cấu trúc dữ liệu khác nhau có thể biểu diễn dữ liệu dưới dạng đồ thị để thống nhất kiểu dạng cho các mục đích khai phá dữ liệu. Tuy nhiên, dữ liệu dạng đồ thị là một dạng dữ liệu phức tạp khi áp dụng các phương pháp khai phá dữ liệu sẽ gặp nhiều khó khăn do hầu hết các công bố khai phá dữ liệu đều có độ phức tạp thời gian không đa thức thậm chí là độ phức tạp hàm mũ. Trong luận án này, nghiên cứu sinh tập trung vào khai phá dữ liệu đồ thị con thường xuyên và phân loại đa nhãn đồ thị. Đối với bài toán khai phá đồ thị con thường xuyên, một vấn đề nổi cộm là xác định đẳng cấu đồ thị con thông thường có độ phức tạp không đa thức. Luận án đã giải quyết khai phá đồ thị con thường xuyên đóng bằng thuật toán PSI-CFSM trong đó vấn đề xác định đẳng cấu đồ thị con trong thời gian đa thức bằng cách áp dụng một số điều kiện ràng buộc về nhãn chuẩn hóa, máy truy cập ngẫu nhiên. Đối với bài toán phân loại đa nhãn, các mô hình phân loại đa nhãn áp dụng lý thuyết Dempster Shafer tăng độ chính xác phân loại và giảm thời gian tính toán không áp dụng được cho biểu diễn dữ liệu đồ thị do đồ thị thiếu biểu diễn dạng véctơ. Luận án thực hiện xây dựng dàn giao khái niệm dựa trên tập đồ thị con thường xuyên đóng của tập dữ liệu đồ thị để từ đó xác định độ đo khoảng cách giữa các đồ thị và dựa vào độ đo khoảng cách này để phân loại đa nhãn cho đồ thị theo lý thuyết Dempster Shafer. Trong luận án, các đề xuất của nghiên cứu sinh về xác định đẳng cấu đồ thị con và xác định độ đo khoảng cách trên dàn giao khái niệm được chứng minh tính đúng đắn và đầy đủ cùng với thực nghiệm chứng tỏ thuật toán PSI-CFSM tối ưu thời gian hơn so với thuật toán gSpan trong khai phá đồ thị con thường xuyên.

**Hướng nghiên cứu tiếp theo**

Dữ liệu đồ thị lớn đang là mục tiêu thách thức hàng đầu hiện nay do các thuật toán khai phá dữ liệu trên đồ thị đa phần là có độ phức tạp thời gian không đa thức hoặc

thậm chí là có độ phức tạp hàm mũ. Dựa trên kết quả rút gọn đối tượng, rút gọn thuộc tính, sinh cây quyết định trên bảng quyết định nhất quán. Nghiên cứu sinh định hướng tiếp tục nghiên cứu về rút gọn đối tượng trên các dữ liệu có cấu trúc phức tạp như đồ thị nhằm mục tiêu giảm không gian lưu trữ và tối ưu thời gian tính toán.

Dữ liệu càng ngày càng tăng liên tục không ngừng do đó khai phá dữ liệu phải đáp ứng trong thời gian thực. Nghiên cứu sinh định hướng giải quyết các bài toán về dữ liệu gia tăng liên tục, áp dụng các kỹ thuật khai phá dữ liệu đã thực hiện thành công trong luận án này lên dữ liệu liên tục tăng trưởng như rút gọn đối tượng, rút gọn thuộc tính, sinh cây quyết định đối với bảng quyết định nhất quán tăng trưởng theo thời gian hay khai phá đồ thị con thường xuyên, phân loại đa nhãn đồ thị trên cơ sở dữ liệu đồ thị tăng trưởng. Dữ liệu đồ thị tăng trưởng có thể kể đến như sự trao đổi chất của các tế bào trong cơ thể, sự phát triển các đoạn gen mang bệnh như ung thư hay sự trao đổi thông tin tức thời các tin nhắn trên mạng xã hội, các thông tin chuyển giao ở tầng mạng trong các cuộc tấn công mạng. Tất cả các dữ liệu tăng trưởng liên tục cần có các phương pháp khai phá tức thời trong thời gian thực là một nhiệm vụ quan trọng trong thời đại bùng nổ thông tin như ngày nay.

# DANH MỤC CÔNG TRÌNH CÔNG BỐ

[1] János Demetrovics, Hoang Minh Quang, Nguyen Viet Anh **and** Vu Duc Thi. "An Optimization of Closed Frequent Subgraph Mining Algorithm". **in**: *Cybernetics and Information Technologies* 17.1 (2017), **pages** 3–15.

[2] János Demetrovics, Hoang Minh Quang, Vu Duc Thi **and** Nguyen Viet Anh. "An Efficient Method to Reduce the Size of Consistent Decision Tables". **in**: *Acta Cybernetica* 23.4 (2018), **pages** 1039–1054. DOI: 10.14232/actacyb.23.4.2018.4.

[3] Hoang Minh Quang **and** Nguyen Ngoc Cuong. "Vấn đề phân loại đa nhãn cho đồ thị". **in**: *Proceeding of the eleventh National Symposium Fundamental and Applied Information Technology Research*. FAIR, Hanoi, Vietnam, 2018, **pages** 567–574.

[4] Hoang Minh Quang, Vu Duc Thi **and** Vu Thi Lan Anh. "Xây dựng cây quyết định từ bảng quyết định nhất quán". **in**: *Proceeding of the tenth National Symposium Fundamental and Applied Information Technology Research*. FAIR, Da Nang, Vietnam, 2017, **pages** 633–640.

[5] Hoang Minh Quang, Vu Duc Thi **and** Pham Quoc Hung. "Một số vấn đề về khai phá đồ thị con thường xuyên đóng". **in**: *Proceeding of the ninth National Symposium Fundamental and Applied Information Technology Research*. FAIR, Can Tho, Vietnam, 2016, **pages** 471–479.

[6] Hoang Minh Quang, Vu Duc Thi **and** Nguyen Ngoc San. "Some algorithms related to consistent decision table". **in**: *Journal of Computer Science and Cybernetics* 33.2 (2017), **pages** 131–142.

[7] Hoang Minh Quang, Vu Duc Thi, Kieu Thu Thuy, Dao Van Tuyet **and** Phan Trung Kien. "Khai phá cây con thường xuyên trên cơ sở dữ liệu weblogs". **in**:

*Proceeding of the eighth National Symposium Fundamental and Applied Information Technology Research*. FAIR, Ha Noi, Vietnam, 2015, **pages** 327–355.

# TÀI LIỆU THAM KHẢO

[1] Charu C Aggarwal, Yuchen Zhao **and** S Yu Philip. "On Clustering Graph Streams." **in**: *SDM*. SIAM. 2010, **pages** 478–489.

[2] Charu Aggarwal, Yan Xie **and** Philip S Yu. "Gconnect: A connectivity index for massive disk-resident graphs". **in**: *Proceedings of the VLDB Endowment* 2.1 (2009), **pages** 862–873.

[3] Rakesh Agrawal, Ramakrishnan Srikant **andothers**. "Fast algorithms for mining association rules". **in**: *Proc. 20th int. conf. very large data bases, VLDB*. **volume** 1215. 1994, **pages** 487–499.

[4] Bahman Bahmani, Ravi Kumar, Mohammad Mahdian **and** Eli Upfal. "Pagerank on an evolving graph". **in**: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2012, **pages** 24–32.

[5] Eugen Barbu, Pierre Heroux, Sebastien Adam **and** Eric Trupin. "Clustering document images using a bag of symbols representation". **in**: *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. IEEE. 2005, **pages** 1216–1220.

[6] Michele Berlingerio, Francesco Bonchi, Björn Bringmann **and** Aristides Gionis. "Mining graph evolution rules". **in**: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2009, **pages** 115–130.

[7] Albert Bifet, Geoff Holmes, Bernhard Pfahringer **and** Ricard Gavaldà. "Mining frequent closed graphs on evolving data streams". **in**: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2011, **pages** 591–599.

[8] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez **and** D-U Hwang. “Complex networks: Structure and dynamics”. **in**: *Physics reports* 424.4 (2006), **pages** 175–308.

[9] Petko Bogdanov, Misael Mongiovì **and** Ambuj K Singh. “Mining heavy subgraphs in time-evolving networks”. **in**: *2011 IEEE 11th International Conference on Data Mining*. IEEE. 2011, **pages** 81–90.

[10] Matthew R Boutell, Jiebo Luo, Xipeng Shen **and** Christopher M Brown. “Learning multi-label scene classification”. **in**: *Pattern recognition* 37.9 (2004), **pages** 1757–1771.

[11] G Burosch, János Demetrovics **and** GOH Katona. “The poset of closures as a model of changing databases”. **in**: *Order* 4.2 (1987), **pages** 127–142.

[12] Jinkun Chen, Jinjin Li, Yaojin Lin, Guoping Lin **and** Zhouming Ma. “Relations of reduction between covering generalized rough sets and concept lattices”. **in**: *Information Sciences* 304 (2015), **pages** 16–27.

[13] Min Chen, Shiwen Mao **and** Yunhao Liu. “Big data: A survey”. **in**: *Mobile networks and applications* 19.2 (2014), **pages** 171–209.

[14] Yun Chi, Yirong Yang **and** Richard R Muntz. “HybridTreeMiner: An efficient algorithm for mining frequent rooted trees and free trees using canonical forms”. **in**: *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*. IEEE. 2004, **pages** 11–20.

[15] Donatello Conte, Pasquale Foggia, Carlo Sansone **and** Mario Vento. “Thirty years of graph matching in pattern recognition”. **in**: *International journal of pattern recognition and artificial intelligence* 18.03 (2004), **pages** 265–298.

[16] Luigi P Cordella, Pasquale Foggia, Carlo Sansone **and** Mario Vento. “A (sub) graph isomorphism algorithm for matching large graphs”. **in**: *IEEE transactions on pattern analysis and machine intelligence* 26.10 (2004), **pages** 1367–1372.

[17] Ma Eugenia Cornejo, Jesús Medina **and** Eloisa Ramírez-Poussa. "Attribute reduction in multi-adjoint concept lattices". **in**: *Information Sciences* 294 (2015), **pages** 41–56.

[18] Bhavana Bharat Dalvi, Meghana Kshirsagar **and** S Sudarshan. "Keyword search on external memory data graphs". **in**: *Proceedings of the VLDB Endowment* 1.1 (2008), **pages** 1189–1204.

[19] Brian A Davey **and** Hilary A Priestley. *Introduction to lattices and order*. Cambridge university press, 2002.

[20] János Demetrovics **and** Vu Duc Thi. "Keys, antikeys and prime attributes". **in**: *Annales Univ. Sci. Budapest, Sect. Comp*. **volume** 8. 1987, **pages** 35–52.

[21] János Demetrovics **and** Vu Duc Thi. "Algorithms for generating an Armstrong relation and inferring functional dependencies in the relational data-model". **in**: *Computers & Mathematics with Applications* 26.4 (1993), **pages** 43–55.

[22] Arthur P Dempster. "The Dempster–Shafer calculus for statisticians". **in**: *International Journal of Approximate Reasoning* 48.2 (2008), **pages** 365–377.

[23] Thierry Denœux. "A k-nearest neighbor classification rule based on Dempster-Shafer theory". **in**: *IEEE transactions on systems, man, and cybernetics* 25.5 (1995), **pages** 804–813.

[24] Thierry Denœux **and** Marie-Hélène Masson. "Evidential reasoning in large partially ordered sets". **in**: *Annals of Operations Research* 195.1 (2012), **pages** 135–161.

[25] Thierry Denœux, Zoulficar Younes **and** Fahed Abdallah. "Representing uncertainty on set-valued variables using belief functions". **in**: *Artificial Intelligence* 174.7 (2010), **pages** 479–499.

[26] Mukund Deshpande, Michihiro Kuramochi, Nikil Wale **and** George Karypis. "Frequent substructure-based approaches for classifying chemical compounds".

**in**: *IEEE Transactions on Knowledge and Data Engineering* 17.8 (2005), **pages** 1036–1050.

[27] Chris HQ Ding, Xiaofeng He, Hongyuan Zha, Ming Gu **and** Horst D Simon. "A min-max cut algorithm for graph partitioning and data clustering". **in**: *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE. 2001, **pages** 107–114.

[28] Mohamed Elhoseny, Ahmed Abdelaziz, Ahmed S Salama, Alaa Mohamed Riad, Khan Muhammad **and** Arun Kumar Sangaiah. "A hybrid model of internet of things and cloud computing to manage big data in health services applications". **in**: *Future generation computer systems* 86 (2018), **pages** 1383–1394.

[29] David Eppstein. "Subgraph isomorphism in planar graphs and related problems". **in**: *SODA*. **volume** 95. 1995, **pages** 632–640.

[30] Bernhard Ganter **and** Rudolf Wille. "Applied lattice theory: Formal concept analysis". **in**: *In General Lattice Theory, G. Grätzer editor, Birkhäuser*. Citeseer. 1997.

[31] Michael R Garey **and** David S Johnson. "Computers and Intractability: An Introduction to the Theory of NP-completeness". **in**: *San Francisco* (1979).

[32] Vijay K Garg, Neeraj Mittal **and** Alper Sen. "Applications of lattice theory to distributed computing". **in**: *ACM SIGACT Notes* 34.3 (2003), **pages** 40–61.

[33] Xin Geng. "Label distribution learning". **in**: *IEEE Transactions on Knowledge and Data Engineering* 28.7 (2016), **pages** 1734–1748.

[34] Nadia Ghamrawi **and** Andrew McCallum. "Collective multi-label classification". **in**: *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM. 2005, **pages** 195–200.

[35] Palash Goyal **and** Emilio Ferrara. “Graph embedding techniques, applications, and performance: A survey”. **in**: *Knowledge-Based Systems* 151 (2018), **pages** 78–94.

[36] Michel Grabisch. “Belief functions on lattices”. **in**: *International Journal of Intelligent Systems* 24.1 (2009), **pages** 76–95.

[37] Jiawei Han, Hong Cheng, Dong Xin **and** Xifeng Yan. “Frequent pattern mining: current status and future directions”. **in**: *Data Mining and Knowledge Discovery* 15.1 (2007), **pages** 55–86.

[38] Jiawei Han, Jian Pei **and** Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[39] Jiawei Han, Jian Pei, Yiwen Yin **and** Runying Mao. “Mining frequent patterns without candidate generation: A frequent-pattern tree approach”. **in**: *Data mining and knowledge discovery* 8.1 (2004), **pages** 53–87.

[40] John E Hopcroft **and** Robert Endre Tarjan. “Isomorphism of planar graphs”. **in**: *Complexity of computer computations*. Springer, 1972, **pages** 131–152.

[41] Tamás Horváth, Jan Ramon **and** Stefan Wrobel. “Frequent subgraph mining in outerplanar graphs”. **in**: *Data Mining and Knowledge Discovery* 21.3 (2010), **pages** 472–508.

[42] Qinghua Hu, Zongxia Xie **and** Daren Yu. “Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation”. **in**: *Pattern recognition* 40.12 (2007), **pages** 3509–3521.

[43] J Huan, W Wang, A Washington, J Prins, R Shah **and** A Tropsha. “Accurate classification of protein structural families using coherent subgraph analysis”. **in**: *Proceedings of the Ninth Pacific Symposium on Biocomputing (PSB)*. 2003, **pages** 411–422.

[44] Jun Huan, Wei Wang **and** Jan Prins. “Efficient mining of frequent subgraphs in the presence of isomorphism”. **in**: *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE. 2003, **pages** 549–552.

[45] Jun Huan, Wei Wang, Jan Prins **and** Jiong Yang. “Spin: mining maximal frequent subgraphs from graph databases”. **in**: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, **pages** 581–586.

[46] Akihiro Inokuchi, Takashi Washio **and** Hiroshi Motoda. “An apriori-based algorithm for mining frequent substructures from graph data”. **in**: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. 2000, **pages** 13–23.

[47] Akihiro Inokuchi, Takashi Washio, Kunio Nishimura **and** Hiroshi Motoda. *A fast algorithm for mining frequent connected subgraphs*. techreport. Technical Report RT0448, IBM Research, Tokyo Research Laboratory, 2002.

[48] Demetrovics Janos, Vu Duc Thi **and** Nguyen Long Giang. “On Finding All Reducts of Consistent Decision Tables”. **in**: *Cybernetics and Information Technologies* 14.4 (2014), **pages** 3–10.

[49] Chuntao Jiang, Frans Coenen **and** Michele Zito. “A survey of frequent subgraph mining algorithms”. **in**: *The Knowledge Engineering Review* 28.01 (2013), **pages** 75–105.

[50] Xiangnan Kong **and** S Yu Philip. “gMLC: a multi-label feature selection framework for graph classification”. **in**: *Knowledge and information systems* 31.2 (2012), **pages** 281–305.

[51] Marzena Kryszkiewicz. “Rough set approach to incomplete information systems”. **in**: *Information sciences* 112.1-4 (1998), **pages** 39–49.

[52] Michihiro Kuramochi **and** George Karypis. “Frequent subgraph discovery”. **in**: *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE. 2001, **pages** 313–320.

[53] Jure Leskovec, Jon Kleinberg **and** Christos Faloutsos. “Graph evolution: Densification and shrinking diameters”. **in**: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), **page** 2.

[54] Guoliang Li, Beng Chin Ooi, Jianhua Feng, Jianyong Wang **and** Lizhu Zhou. “EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data”. **in**: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM. 2008, **pages** 903–914.

[55] Xian-Tong LI, Jian-Zhong LI **and** Hong GAO. “An Efficient Frequent Subgraph Mining Algorithm”. **in**: *Journal of Software* 10 (2007), **page** 011.

[56] Min Liu, Mingwen Shao, Wenxiu Zhang **and** Cheng Wu. “Reduction method for concept lattices based on rough set theory and its application”. **in**: *Computers & Mathematics with Applications* 53.9 (2007), **pages** 1390–1410.

[57] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh **and** Angela H Byers. “Big data: The next frontier for innovation, competition, and productivity”. **in**: (2011).

[58] Brendan D McKay **andothers**. *Practical graph isomorphism*. Department of Computer Science, Vanderbilt University Tennessee, US, 1981.

[59] Ju-Sheng Mi, Wei-Zhi Wu **and** Wen-Xiu Zhang. “Approaches to knowledge reduction based on variable precision rough set model”. **in**: *Information sciences* 159.3 (2004), **pages** 255–272.

[60] Fan Min, Huaping He, Yuhua Qian **and** William Zhu. “Test-cost-sensitive attribute reduction”. **in**: *Information Sciences* 181.22 (2011), **pages** 4928–4942.

[61] Bernard Monjardet. “The presence of lattice theory in discrete problems of mathematical social sciences. Why”. **in**: *Mathematical Social Sciences* 46.2 (2003), **pages** 103–144.

[62] Viet Anh Nguyen **and** Akihiro Yamamoto. “Learning from graph data by putting graphs on the lattice”. **in**: *Expert Systems with Applications* 39.12 (2012), **pages** 11172–11182.

[63] Rickard Nyman, Sujit Kapadia, David Tuckett, David Gregory, Paul Ormerod **and** Robert Smith. “News and narratives in financial systems: exploiting big data for systemic risk assessment”. **in**: (2018).

[64] Zdzislaw Pawlak. “Rough sets”. **in**: *International Journal of Computer & Information Sciences* 11.5 (1982), **pages** 341–356.

[65] Zdzislaw Pawlak. “Rough sets and intelligent data analysis”. **in**: *Information sciences* 147.1 (2002), **pages** 1–12.

[66] Zdzislaw Pawlak, Jerzy Grzymala-Busse, Roman Slowinski **and** Wojciech Ziarko. “Rough sets”. **in**: *Communications of the ACM* 38.11 (1995), **pages** 88–95.

[67] Zdzislaw Pawlak **and** Andrzej Skowron. “Rough sets and Boolean reasoning”. **in**: *Information sciences* 177.1 (2007), **pages** 41–73.

[68] Yuhua Qian **and** Jiye Liang. “Combination entropy and combination granulation in rough set theory”. **in**: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 16.02 (2008), **pages** 179–193.

[69] Yuhua Qian, Jiye Liang, Witold Pedrycz **and** Chuangyin Dang. “Positive approximation: an accelerator for attribute reduction in rough set theory”. **in**: *Artificial Intelligence* 174.9-10 (2010), **pages** 597–618.

[70] Jesse Read, Bernhard Pfahringer **and** Geoff Holmes. “Multi-label classification using ensembles of pruned sets”. **in**: *2008 Eighth IEEE International Conference on Data Mining*. IEEE. 2008, **pages** 995–1000.

[71] Jesse Read, Bernhard Pfahringer, Geoff Holmes **and** Eibe Frank. “Classifier chains for multi-label classification”. **in**: *Machine learning* 85.3 (2011), **pages** 333–359.

[72] Ronald C Read **and** Derek G Corneil. “The graph isomorphism disease”. **in**: *Journal of Graph Theory* 1.4 (1977), **pages** 339–363.

[73] John E Savage. “Models of computation”. **in**: *Exploring the Power of Computing* (1998).

[74] Glenn Shafer **andothers**. *A mathematical theory of evidence*. **volume** 1. Princeton university press Princeton, 1976.

[75] Andrzej Skowron **and** Cecylia Rauszer. “The discernibility matrices and functions in information systems”. **in**: *Intelligent Decision Support*. Springer, 1992, **pages** 331–362.

[76] Dominik Ślezak. “Approximate entropy reducts”. **in**: *Fundamenta informaticae* 53.3-4 (2002), **pages** 365–390.

[77] N Talukder **and** MJ Zaki. “A distributed approach for graph mining in massive networks”. **in**: *Data Mining and Knowledge Discovery* (2016), **pages** 1–29.

[78] Vu Duc Thi. “The minimal keys and antikeys”. **in**: *Acta Cybernetica* 7.4 (1986), **pages** 361–371.

[79] Vu Duc Thi **and** Nguyen Long Giang. “A Method to Construct Decision Table from Relation Scheme”. **in**: *Cybernetics and Information Technologies* 11.3 (2011), **pages** 32–41.

[80] Vu Duc Thi **and** Nguyen Long Giang. “Some Problems concerning Condition Attributes and Reducts in Decision Tables”. **in**: *Proceeding of the fifth National Symposium Fundamental and Applied Information Technology Research*. FAIR, Dong Nai, Vietnam, 2012, **pages** 142–152.

[81] Lini T Thomas, Satyanarayana R Valluri **and** Kamalakar Karlapalem. "Margin: Maximal frequent subgraph mining". **in**: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4.3 (2010), **page** 10.

[82] Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris **and** Ioannis P Vlahavas. "Multi-Label Classification of Music into Emotions." **in**: *ISMIR*. **volume** 8. 2008, **pages** 325–330.

[83] Grigorios Tsoumakas **and** Ioannis Katakis. "Multi-label classification: An overview". **in**: *Dept. of Informatics, Aristotle University of Thessaloniki, Greece* (2006).

[84] Julian R Ullmann. "An algorithm for subgraph isomorphism". **in**: *Journal of the ACM (JACM)* 23.1 (1976), **pages** 31–42.

[85] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski **and** Hendrik Blockeel. "Decision trees for hierarchical multi-label classification". **in**: *Machine Learning* 73.2 (2008), **pages** 185–214.

[86] Takashi Washio **and** Hiroshi Motoda. "State of the art of graph-based data mining". **in**: *Acm Sigkdd Explorations Newsletter* 5.1 (2003), **pages** 59–68.

[87] Wei Wei, Junhong Wang, Jiye Liang, Xin Mi **and** Chuangyin Dang. "Compacted decision tables based attribute reduction". **in**: *Knowledge-Based Systems* 86 (2015), **pages** 261–277.

[88] Xifeng Yan **and** Jiawei Han. "gspan: Graph-based substructure pattern mining". **in**: *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE. 2002, **pages** 721–724.

[89] Xifeng Yan **and** Jiawei Han. "CloseGraph: mining closed frequent graph patterns". **in**: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2003, **pages** 286–295.

[90] Xifeng Yan, Philip S Yu **and** Jiawei Han. “Graph indexing: a frequent structure-based approach”. **in**: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM. 2004, **pages** 335–346.

[91] Xifeng Yan, Feida Zhu, Jiawei Han **and** Philip S Yu. “Searching substructures with superimposed distance”. **in**: *22nd International Conference on Data Engineering (ICDE’06)*. IEEE. 2006, **pages** 88–88.

[92] Yiyu Yao **and** Yan Zhao. “Attribute reduction in decision-theoretic rough set models”. **in**: *Information sciences* 178.17 (2008), **pages** 3356–3373.

[93] Yiyu Yao **and** Yan Zhao. “Discernibility matrix simplification for constructing attribute reducts”. **in**: *Information sciences* 179.7 (2009), **pages** 867–882.

[94] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton **and** Jure Leskovec. “Graph convolutional neural networks for web-scale recommender systems”. **in**: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. 2018, **pages** 974–983.

[95] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton **and** Jure Leskovec. “Hierarchical graph representation learning with differentiable pooling”. **in**: *Advances in Neural Information Processing Systems*. 2018, **pages** 4800–4810.

[96] Ronghui You, Zihan Zhang, Yi Xiong, Fengzhu Sun, Hiroshi Mamitsuka **and** Shanfeng Zhu. “GOLabeler: improving sequence-based large-scale protein function prediction by learning to rank”. **in**: *Bioinformatics* 34.14 (2018), **pages** 2465–2473.

[97] Zoulficar Younes, Fahed Abdallah **and** Thierry Denœux. “An evidence-theoretic k-nearest neighbor rule for multi-label classification”. **in**: *International Conference on Scalable Uncertainty Management*. Springer. 2009, **pages** 297–308.

[98] Zoulficar Younes, Thierry Denœux **andothers**. “Evidential multi-label classification approach to learning from data with imprecise labels”. **in**: *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer. 2010, **pages** 119–128.

[99] Min-Ling Zhang **and** Zhi-Hua Zhou. “A k-nearest neighbor based algorithm for multi-label classification”. **in**: *2005 IEEE international conference on granular computing*. **volume** 2. IEEE. 2005, **pages** 718–721.

[100] Kai Zheng, Jie Hu, Zhenfei Zhan, Jin Ma **and** Jin Qi. “An enhancement for heuristic attribute reduction algorithm in rough set”. **in**: *Expert Systems with Applications* 41.15 (2014), **pages** 6748–6754.

[101] Zhenyu Zhou, Caixia Gao, Chen Xu, Yan Zhang, Shahid Mumtaz **and** Jonathan Rodriguez. “Social big-data-based content dissemination in Internet of vehicles”. **in**: *IEEE Transactions on Industrial Informatics* 14.2 (2018), **pages** 768–777.