ĐẶNG TIẾN ĐẠT

TOÁN ỨNG DỤNG

2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Đặng Tiến Đạt

MỘT SỐ THUẬT TOÁN TÌM KIẾM CỘNG ĐỒNG MẠNG THÔNG QUA TỐI ƯU HOÁ HÀM MODULARITY

LUẬN VĂN THẠC SĨ TOÁN HỌC

Hà Nội - 2024

BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Đặng Tiến Đạt

MỘT SỐ THUẬT TOÁN TÌM KIẾM CỘNG ĐỒNG MẠNG THÔNG QUA TỐI ƯU HOÁ HÀM MODULARITY

LUẬN VĂN THẠC SĨ TOÁN HỌC

Ngành: Toán ứng dụng

Mã số: 8 46 01 12

NGƯỜI HƯỚNG DẪN KHOA HỌC :
1. TS. Đỗ Duy Hiếu
2. PGS. TS. Vũ Thế Khôi

Hà Nội - 2024

LỜI CAM ĐOAN

Tôi xin cam đoan đề tài nghiên cứu trong luận văn này là công trình nghiên cứu của tôi dựa trên những tài liệu, số liệu do chính tôi tự tìm hiểu và nghiên cứu. Chính vì vậy, các kết quả nghiên cứu đảm bảo trung thực và khách quan nhất. Đồng thời, kết quả này chưa từng xuất hiện trong bất cứ một nghiên cứu nào. Các số liệu, kết quả nêu trong luận văn là trung thực nếu sai tôi hoàn chịu trách nhiệm trước pháp luật.

Tác giả luận văn

Đặng Tiến Đạt

LỜI CẢM ƠN

Trước tiên, tôi xin bày tỏ lòng biết ơn sâu sắc đến **TS. Đỗ Duy Hiếu**, người đã tận tình hướng dẫn, động viên và chia sẻ những kiến thức quý báu, giúp tôi hoàn thành luận văn một cách tốt nhất. Tôi cũng xin gửi lời cảm ơn chân thành đến **PGS. TSKH. Phan Thị Hà Dương**, người đã không chỉ truyền đạt kiến thức sâu sắc về thuật toán mà còn truyền cảm hứng đam mê nghiên cứu, giúp tôi luôn kiên trì và phấn đấu trên con đường học thuật. Tôi xin được gửi lời cảm ơn tới **PGS. TS. Vũ Thế Khôi**, người đã truyền đạt cho tôi những kiến thức quan trọng về kỹ năng công bố khoa học và liêm chính học thuật.

Tôi chân thành cảm ơn các thầy cô tại **Viện Toán học**, những người đã trang bị cho tôi nền tảng kiến thức vững chắc từ các môn học, tạo điều kiện để tôi hoàn thành tốt luận văn này.

Tôi xin gửi lời chân thành cảm ơn tới **Quỹ Đổi mới sáng tạo Vingroup (VinIF)** đã cấp học bổng Thạc sĩ trong nước cho tôi vào năm 2022, việc được nhận học bổng đã giúp tôi có thêm nhiều đồng lực để học tập và nghiên cứu trong khoảng thời gian theo học tại Viện Toán học.

Cuối cùng, tôi xin gửi lời cảm ơn đến Ban Lãnh đạo cùng Phòng Đào tạo của **Viện Toán học, Học viện Khoa học và Công Nghệ**, những người đã hỗ trợ và tạo điều kiện thuận lợi cho tôi trong suốt quá trình học tập và nghiên cứu, để tôi có thể đạt được thành quả như ngày hôm nay.

Tác giả luận văn

Đặng Tiến Đạt

Mục lục

DANH MỤC CÁC KÝ HIỆU, CHỮ CÁI VIẾT TẮT

Ký hiệu	Định nghĩa
$G(V, E)$	Một đồ thị G với tập đỉnh V và tập cạnh E
A	Ma trận kề của đồ thị G
d_i	Bậc của đỉnh i
d_i^{in}	Bậc vào của đỉnh i
d_i^{out}	Bậc ra của đỉnh i
P	Ma trận xác suất chuyển trên đô thị
ϕ_i	Xác suất bước đi ngẫu nhiên đạt tại đỉnh i tại thời điểm dừng
ϕ	Phân phối dừng của bước đi ngẫu nhiên: $\phi = (\phi_1, \phi_2, ..., \phi_n)$
$\mathcal{P}$	là một cách phân chia cộng đồng $\mathcal{P} = \{C_1, C_2, \cdots, C_K\}$
N_{C_k}	Số lượng đỉnh trong cộng đồng C_k
$\lvert C_k \cap C_h \rvert$	Số lượng đỉnh có mặt trong cả hai cộng đồng C_k và C_h
$\lvert E_C^{in} \rvert$	Tổng số cạnh trong cộng đồng C
$\lvert E_C^{out} \rvert$	Tổng số cạnh giữa các đỉnh trong cộng đồng C và các đỉnh ngoài cộng đồng C
$D(C)$	Tổng bậc của các đỉnh trong cộng đồng C
Q_u	Hàm modularity cho đồ thị vô hướng
Q_d	Hàm modularity cho đồ thị có hướng
Q_f	Hàm modularity tổng quát có hệ số phân giải cho đồ thị vô hướng
Q_p	Hàm modularity đề xuất cho đồ thị có hướng
Q_u^C	Giá trị modularity Q_u trên cộng đồng C
Q_f^C	Giá trị modularity Q_f trên cộng đồng C

Trong luận văn này, các cặp cụm từ "mạng" và "đồ thị"; cụm từ "cộng đồng" và "cụm" cũng được sử dụng với ý nghĩa như nhau.

Danh sách bảng

MỞ ĐẦU

Lý do chọn đề tài

Bài toán phát hiện cộng đồng mạng đặc biệt quan trọng trong khoa học máy tính cũng như trong nhiều ngành khoa học khác. Vì vậy, bài toán này đã được rất nhiều nhà toán học, tin học, vật lý… trên thế giới quan tâm nghiên cứu. Đặc biệt là hướng tiếp cận tối ưu hoá hàm modularity như: Thuật toán Louvain [1], Thuật toán Leiden [2], Phương pháp Phổ của Newman [3]…

Ở Việt Nam, cũng có một số nhóm quan tâm nghiên cứu bài toán tìm kiếm cộng đồng mạng như nhóm của PGS. TSKH. Phan Thị Hà Dương và TS. Đỗ Duy Hiếu – Viện Toán học. Nhóm của GS. TSKH. Nguyễn Đông Yên – Viện Toán học (nghiên cứu về thuật toán K-Means).

Modularity là một thước đo quan trọng trong việc đánh giá chất lượng phân cụm trong mạng, phản ánh không chỉ mức độ gắn kết bên trong từng cộng đồng mà còn mức độ tách biệt giữa các cộng đồng khác nhau. Giá trị modularity càng cao, cấu trúc phân chia cộng đồng càng rõ ràng, cho thấy mạng có tính cộng đồng mạnh mẽ. Chính vì vai trò quan trọng này, việc phát triển các định nghĩa mới về modularity và nghiên cứu các thuật toán tối ưu hóa modularity để tìm kiếm cộng đồng mạng đã thu hút sự quan tâm và nghiên cứu sâu rộng trong lĩnh vực này.

Mục đích nghiên cứu

- Tìm hiểu về tìm kiếm cộng đồng mạng thông qua tối ưu modularity: các khái niệm về cộng đồng mạng, hàm modularity, trạng thái dừng, quá trình bước đi ngẫu nhiên, các bài toán tối ưu và một số thuật toán tối ưu.

- Tìm hiểu về các thuật toán tối ưu hóa modularity cục bộ: thuật toán Louvain và thuật toán Leiden.
- Xây dựng hàm modularity thông qua bước đi ngẫu nhiên.
- Tìm hiểu các thuật toán tối ưu modularity toàn cục : phương pháp phổ của Newman cho đồ thị vô hướng và đề xuất phương pháp mới cho đồ thị có hướng.

Nội dung nghiên cứu

- Đối tượng nghiên cứu : Cộng đồng mạng trong đồ thị.
- Phạm vi nghiên cứu : Hàm modularity cho đồ thị vô hướng và có hướng, các thuật toán phương pháp phổ tối ưu hàm modularity toàn cục và các thuật toán tối ưu cục bộ.

Cơ sở khoa học và tính thực tiễn của đề tài

Luận văn được xây dựng dựa trên các cơ sở khoa học sau:

- Lý thuyết đồ thị: các đại lượng cơ bản của đồ thị, quá trình ngẫu nhiên trên đồ thị, định nghĩa cộng đồng mạng và các thuật toán tìm kiếm cộng đồng mạng.
- Hàm modularity và giải bài toán tối ưu: các hàm modularity đánh giá chất lượng cộng đồng, phương pháp nhân tử Lagrange.
- Đánh giá kết quả: các mô hình đồ thị ngẫu nhiên và các đại lượng đánh giá chất lượng cộng đồng.

Tính thực tiễn của đề tài được thể hiện qua:

- Thuật toán phát hiện cộng đồng có thể được áp dụng để phân tích cấu trúc nhóm trong nhiều lĩnh vực, như mạng xã hội và sinh học. Trong mạng xã hội, thuật toán giúp xác định các cộng đồng có chung sở thích

hoặc nhóm ảnh hưởng trên các nền tảng như Facebook, Twitter. Trong sinh học, nó hỗ trợ phân tích mạng lưới gen và protein để tìm ra các nhóm chức năng tương tác trong sinh học phân tử.

- Các thuật toán tìm kiếm cộng đồng mạng bằng cách tối ưu hóa hàm modularity đưa ra kết quả là phát hiện được các cộng đồng tốt và thời gian chạy nhanh, có thể ứng dụng vào các đồ thị phức tạp, lớn trong thực tế.

Những đóng góp của luận văn

- Trình bày về các thuật toán tìm kiếm cộng đồng mạng của đồ thị vô hướng thông qua tối ưu hàm modularity cục bộ: thuật toán Louvain và thuật toán Leiden
- Trình bày phương pháp phổ của Newman tối ưu modularity toàn cục cho đồ thị vô hướng.
- Đề xuất hàm modularity cho đồ thị có hướng dựa trên quá trình bước đi ngẫu nhiên và phương pháp phổ để tìm kiếm cộng đồng mạng cho đồ thị có hướng.

Bố cục luận văn

Luận văn được tổ chức thành ba chương chính như sau:

- Chương 1: Trình bày kiến thức chuẩn bị: Trình bày về mạng, cộng đồng mạng, tìm kiếm cộng đồng mạng. Trình bày kiến thức liên quan: bước đi ngẫu nhiên, modularity, trạng thái dừng, ...
- Chương 2: Trình bày về các thuật toán tối ưu modularity cục bộ: thuật toán Louvain và thuật toán Leiden.
- Chương 3: Trình bày các thuật toán tối ưu hàm modularity toàn cục: thuật toán phương pháp phổ của Newman cho đồ thị vô hướng, hàm modularity đề xuất và thuật toán đề xuất cho đồ thị có hướng.

Chương 1

Kiến thức chuẩn bị

1.1. Các đại lượng cơ bản của đồ thị

Đồ thị $G = (V, E)$ được định nghĩa bởi tập đỉnh V và tập cạnh E. Trong trường hợp đồ thị đơn, giữa hai đỉnh i và j thuộc V chỉ có duy nhất một cạnh nối trực tiếp. Ngoài ra, những cạnh có điểm đầu và điểm cuối trùng nhau, tức là xuất phát và kết thúc tại cùng một đỉnh, được gọi là khuyên.

Các khái niệm cơ bản dưới đây sẽ được sử dụng xuyên suốt trong luận văn này.

Định nghĩa 1.1.1 (Đồ thị có hướng [4])**.** *Đồ thị có hướng $G = (V, E)$ là một bộ ba gồm một tập đỉnh V, một tập cạnh E, và một hàm ánh xạ mỗi cạnh thành một cặp đỉnh có thứ tự. Đỉnh đầu tiên của cặp có thứ tự là đuôi của cạnh, và đỉnh thứ hai là đầu; cùng nhau, hai đỉnh tạo thành điểm đầu cuối. Một cạnh được xác định là một cạnh từ đuôi của nó đến đầu của nó.*

Những cạnh trong đồ thị có hướng được gọi là cạnh có hướng, có thể được biểu diễn bởi những đường thẳng có mũi tên chỉ hướng. Trong khi đó, đồ thị vô hướng được xác định khi cạnh không có hướng, tức mối quan hệ hai chiều, từ đây ta có định nghĩa của đồ thị vô hướng như sau:

Định nghĩa 1.1.2 (Đồ thị vô hướng)**.** *Một đồ thị vô hướng $G = (V, E)$ là một bộ ba gồm một tập đỉnh V, một tập cạnh E, và một hàm ánh xạ mỗi cạnh tới một cặp đỉnh không có thứ tự.*

Ma trận kề là một công cụ toán học phổ biến dùng để biểu diễn đồ thị. Nó cung cấp cách thức mô tả sự kết nối giữa các đỉnh trong đồ thị thông qua các

cạnh. Cụ thể, ma trận kề của một đồ thị được định nghĩa như sau:

Định nghĩa 1.1.3 (Ma trận kề của đồ thị [5]). *Ma trận kề A của đồ thị đơn $G = (V, E)$ là một ma trận vuông kích thước $n \times n$, trong đó $n = |V|$ là số đỉnh của đồ thị. Các phần tử của ma trận A được xác định như sau:*

$$A_{ij} = \begin{cases} 1, & \text{nếu có cạnh từ đỉnh } i \text{ đến đỉnh } j, \quad \forall i, j \in 1, \ldots, |V|. \\ 0, & \text{trong trường hợp ngược lại.} \end{cases}$$

Ngoài đồ thị đơn, trong thực tế tồn tại nhiều đồ thị với sự liên kết giữa các đỉnh không chỉ là các cạnh đơn mà có thể nhiều cạnh hoặc trọng số của cạnh, đồ thị có trọng số được định nghĩa như sau:

Định nghĩa 1.1.4 (Đồ thị có trọng số [5]). *Một đồ thị G_w có trọng số được biểu diễn thông qua 3 đại lượng $G_w = (V, E, W)$ trong đó W là ma trận biểu diễn trọng số cho mỗi cạnh với W_{ij} là trọng số của cạnh nối giữa hai đỉnh i, j.*

Trong lý thuyết đồ thị, một khái niệm quan trọng là bậc của một đỉnh, thể hiện số lượng kết nối của đỉnh đó với các đỉnh khác trong đồ thị. Để hiểu rõ hơn về mức độ liên kết của các đỉnh trong đồ thị, ta định nghĩa bậc của đỉnh như sau:

Định nghĩa 1.1.5 (Bậc của đỉnh trong đồ thị [5]). *Bậc của đỉnh i trong đồ thị G được tính dựa trên tổng các trọng số của các cạnh nối từ đỉnh i đến các đỉnh khác trong đồ thị. Đối với từng loại đồ thị, bậc của mỗi đỉnh được xác định cụ thể như sau:*

- ***Đồ thị vô hướng, không trọng số**: Bậc của đỉnh i, ký hiệu là d_i, được tính bằng tổng số cạnh nối đến đỉnh i:*

  $$d_i = \sum_{j=1}^{n} A_{ij} = \sum_{j=1}^{n} A_{ji},$$

  trong đó A_{ij} là phần tử của ma trận kề A, biểu thị sự tồn tại của cạnh giữa hai đỉnh i và j.

- ***Đồ thị có hướng, không trọng số**: Do cạnh có hướng, bậc của đỉnh i được chia thành hai loại:*

– ***Bậc vào*** d_i^{in}*: là tổng số cạnh đi vào đỉnh i,*

$$d_i^{in} = \sum_{j=1}^{n} A_{ji},$$

– ***Bậc ra*** d_i^{out}*: là tổng số cạnh đi ra từ đỉnh i,*

$$d_i^{out} = \sum_{j=1}^{n} A_{ij}.$$

- ***Đồ thị có trọng số****: Bậc của đỉnh i được tính tương tự, nhưng thay các giá trị* A_{ij} *bằng trọng số của cạnh giữa hai đỉnh i và j. Cụ thể:*

$$d_i = \sum_{j=1}^{n} w_{ij},$$

trong đó w_{ij} *là trọng số của cạnh giữa i và j.*

Từ đó, chúng ta sẽ định nghĩa ma trận bậc cho đồ thị vô hướng và có hướng như sau:

Định nghĩa 1.1.6 (Ma trận bậc của đồ thị [5])**.**

- *Trong đồ thị vô hướng, ma trận bậc D là ma trận đường chéo, trong đó phần tử* D_{ii} *là bậc của đỉnh i, tức* $D_{ii} = d_i$*.*

$$D = \begin{bmatrix} d_1 & 0 & ... & 0 \\ 0 & d_2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & d_n \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{1.1.1}$$

- *Trong đồ thị có hướng, ma trận bậc đi ra* D^{out} *là một ma trận đường chéo, trong đó phần tử* D_{ii}^{out} *là bậc đi ra của đỉnh i, tức là* $D_{ii}^{out} = d_i^{out}$*.*

$$D^{out} = \begin{bmatrix} d_1^{out} & 0 & ... & 0 \\ 0 & d_2^{out} & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & d_n^{out} \end{bmatrix} \in \mathbb{R}^{n \times n}. \tag{1.1.2}$$

Kể từ thời điểm này trong luận văn, với trường hợp đồ thị có hướng, ma trận bậc sẽ được hiểu là ma trận bậc đi ra.

Tiếp theo, chúng tôi sẽ giới thiệu về ma trận Laplacian. Ma trận Laplacian trong đồ thị là một công cụ quan trọng trong lý thuyết đồ thị, giúp biểu diễn các thuộc tính kết nối và cấu trúc của đồ thị, từ đó hỗ trợ phân tích các đặc điểm như tính liên thông và phát hiện cộng đồng trong đồ thị.

Định nghĩa 1.1.7 (Ma trận Laplacian [5])**.** *Ma trận Laplacian L của đồ thị G được đinh nghĩa là:*

$$L = D - A, \tag{1.1.3}$$

trong đó A, D lần lượt là ma trận kề và ma trận bậc của đồ thị G.

1.2. Quá trình ngẫu nhiên trên đồ thị

Bước đi ngẫu nhiên trên đồ thị là một quá trình ngẫu nhiên mô tả sự di chuyển trên đồ thị, bắt đầu từ một đỉnh ban đầu, ở mỗi bước, ta chọn ngẫu nhiên một cạnh nối với đỉnh hiện tại và di chuyển đến đỉnh ở đầu kia của cạnh đó, sau đó tiếp tục lặp lại quá trình này.

Từ đó, chúng ta có thể định nghĩa về ma trận xác suất chuyển trên đồ thị sau như sau (áp dụng cho cả đồ thị vô hướng và có hướng):

Định nghĩa 1.2.1 (Ma trận xác suất chuyển trên đồ thị [5])**.** *Đồ thị $G = (V, E)$ có ma trận kề $A = (A_{ij})_{n\times n}$ và ma trận bậc D, gọi P_{ij} là xác suất di chuyển từ đỉnh i sang đỉnh j, xác suất này được xác định như sau:*

$$P_{ij} = \frac{A_{ij}}{D_{ii}} = \frac{A_{ij}}{\sum_{j=1}^{n} A_{ij}} \tag{1.2.1}$$

Công thức trên có thể viết dưới dạng ma trận:

$$P = D^{-1}A = (P_{ij})_{n\times n}. \tag{1.2.2}$$

Từ công thức 1.2.1, ta kí hiệu P_{ij}^t là xác suất đỉnh i di chuyển tới đỉnh j sau t bước; dưới dạng ma trận, ta có $P^t = (P_{ij}^t)_{n\times n}$ là ma trận xác suất chuyển với độ dài t bước.

Ta thấy, xác suất chuyển là xác suất có điều kiện khi từ đỉnh hiện tại, ta chuyển sang đỉnh khác với xác suất được tính dựa trên bậc của đỉnh hiện tại. Bên cạnh xác suất chuyển này, xác suất để bước đi rơi vào các đỉnh tại thời điểm ban đầu và sau t bước cũng được quan tâm. Xem xét thời điểm ban đầu, bước đi có thể bắt đầu từ đỉnh i với xác suất p_i, bắt đầu tại đỉnh j với xác suất p_j, ... Từ đây ta kí hiệu xác suất để sau t bước, bước đi rơi vào một đỉnh i cụ thể là $p_i^{(t)}$ (đặt $p^{(0)} = p$, tổng hợp lại mọi đỉnh, ta được $p^{(t)} = (p_1^{(t)}, p_2^{(t)}, ..., p_n^{(t)})$ là vector xác suất. Dễ thấy

$$\sum_{i=1}^{n} p_i^{(t)} = 1, \tag{1.2.3}$$

Đặc biệt hơn, xác suất $p_i^{(t)}$ có thể được tổng hợp từ các $p_j^{(t-1)}$:

$$p_i^{(t)} = \sum_j P_{ji} p_j^{(t-1)}, \tag{1.2.4}$$

hay $p^{(t)} = p^{(t-1)}P$ dưới dạng ma trận. Từ đây, ta biểu diễn được $p^{(t)}$ theo $p^{(0)}$ và P^t:

$$p^{(t)} = p^{(0)}P^t, \tag{1.2.5}$$

Một trong các đại lượng quan trọng thể hiện xác suất bước đi ngẫu nhiên rơi vào các đỉnh chính là trạng thái dừng.

Định nghĩa 1.2.2 (Trạng thái dừng Ergodic [6]). *Phân phối $\pi = (\pi_1, \pi_2, ..., \pi_S) = (\pi_i)_{i \in S}$ trên không gian trạng thái S được gọi là phân phối dừng Ergodic với ma trận chuyển P khi với mọi trạng thái j, ta có:*

$$\lim_{t \to \infty} p_j^{(t)} = \pi_j, \tag{1.2.6}$$

Đặc biệt với mọi cặp trạng thái i, j, ta cũng thu được xác suất chuyển giới hạn:

$$\lim_{t \to \infty} P_{ij}^t = \pi_j. \tag{1.2.7}$$

Nếu phân phối π tồn tại thì sẽ là nghiệm của phương trình:

$$\pi = \pi P, \tag{1.2.8}$$

Để trạng thái dừng của bước đi ngẫu nhiên trên đồ thị tồn tại, cần thỏa mãn một số điều kiện cụ thể [6]:

- **Đối với đồ thị vô hướng**: Đồ thị phải liên thông, nghĩa là từ bất kỳ đỉnh nào cũng có thể đi đến mọi đỉnh khác. Hơn nữa, một điều kiện quan trọng là ước chung lớn nhất (UCLN) của số bước để một đỉnh i quay lại chính nó phải bằng 1, tức là $UCLN(\{t|t \in \mathbb{N}^*, P_{ii}^t > 0\}) = 1$. Điều kiện này ngăn quá trình bước đi ngẫu nhiên rơi vào một chu kỳ cố định có độ dài lớn hơn 1, giúp đảm bảo rằng trạng thái dừng có thể đạt được với mọi t đủ lớn.
- **Đối với đồ thị có hướng**: Đồ thị cần liên thông mạnh, tức là có thể di chuyển từ mọi đỉnh đến tất cả các đỉnh khác theo hướng của các cạnh. Tương tự, UCLN của số bước để một đỉnh i quay lại chính nó cũng phải bằng 1, tức là $UCLN(\{t|t \in \mathbb{N}^*, P_{ii}^t > 0\}) = 1$.

1.3. Mạng và tìm kiếm cộng đồng mạng

Mạng là cấu trúc có thể được hình dung một cách cơ bản nhất như một tập hợp các điểm được nối với nhau thành từng cặp bằng các đường thẳng. Trong ngôn ngữ toán học, các điểm này được gọi là đỉnh (hoặc nút), và các đường nối giữa chúng được gọi là cạnh. Trong bối cảnh cấu trúc dữ liệu và thuật toán, đồ thị đại diện cho một cấu trúc dữ liệu phi tuyến, mô tả mối quan hệ giữa các đỉnh thông qua tập hợp các cạnh.

Đồ thị được ứng dụng rộng rãi trong thực tế, đặc biệt trong nghiên cứu mối quan hệ giữa các đối tượng như mạng xã hội, mạng protein, và mạng lưới sân bay... Ví dụ, mạng Internet có thể được xem như một tập hợp các máy tính kết nối với nhau, trong khi xã hội con người là một mạng lưới các cá nhân gắn kết qua các mối quan hệ. Nghiên cứu đồ thị có thể tập trung vào đặc điểm của các đỉnh, như hoạt động của máy tính, hoặc các cạnh, như cách thức tương tác diễn ra. Một khía cạnh quan trọng khác là cấu trúc kết nối giữa các đỉnh, quyết định đến đặc tính của mạng, khả năng lan truyền thông tin, tính bền vững trước các tấn công, và sự hình thành của các cấu trúc con.

Cộng đồng mạng: là nhóm các đỉnh có liên kết chặt chẽ với nhau hơn khi so với các đỉnh bên ngoài. Những mối quan hệ này có thể được đo lường qua

các chỉ số như số đỉnh chung, số cạnh kết nối, số đường đi ngắn nhất, hoặc độ tương đồng dựa trên các đặc tính cụ thể. Tùy vào cách định nghĩa mối quan hệ "tốt" giữa các đỉnh, các cộng đồng sẽ được phát hiện và mô tả khác nhau, dựa trên tiêu chí đo lường này.

Tìm kiếm cộng đồng mạng: là quá trình xác định và phân nhóm các đỉnh trong một đồ thị thành các cộng đồng hoặc cụm, sao cho các đỉnh trong cùng một nhóm có kết nối chặt chẽ với nhau hơn so với các đỉnh thuộc nhóm khác.

Cộng đồng mạng thường phân thành hai loại: tách biệt, nơi mỗi đỉnh chỉ thuộc một cộng đồng, và chồng chéo, nơi một đỉnh có thể thuộc nhiều cộng đồng. Trong phạm vi luận văn này, chúng tôi tập trung vào phân tích cộng đồng tách biệt.

Do số lượng và tính đa dạng lớn của các đồ thị, các thuật toán nghiên cứu trong lĩnh vực này liên tục được phát triển. Trong số đó, một số thuật toán truyền thống được nhiều người quan tâm là:

- **Thuật toán phân vùng đồ thị [7]**: Đây là kỹ thuật chia đồ thị thành một số cụm nhất định, với kích thước của mỗi cụm đã được xác định trước. Thuật toán này thường được áp dụng để tối ưu hóa các vấn đề liên quan đến phân bổ tài nguyên và xử lý dữ liệu phân tán.

- **Thuật toán phân cụm phân cấp [7]**: Thuật toán này dựa trên giả thuyết rằng mỗi cộng đồng có thể được coi là sự hợp thành của nhiều cộng đồng nhỏ hơn ở các cấp độ khác nhau, tạo ra một cấu trúc cộng đồng đa cấp, phản ánh sự phân tầng của đồ thị. Hai yếu tố quan trọng của thuật toán này là tiêu chí hợp/tách cụm và cách chọn lát cắt tối ưu.

- **Phân cụm dựa trên phương pháp phổ [7]**: Phương pháp này bao gồm các kỹ thuật sử dụng tính toán vector riêng và giá trị riêng để phân chia các điểm dữ liệu thành các cộng đồng. Một thuật toán quan trọng được đề cập trong báo cáo này là tối ưu hóa modularity dựa trên phương pháp phổ, áp dụng cho cả đồ thị có hướng và vô hướng.

- **Thuật toán dựa trên Louvain** [8]: Thuật toán Louvain là một phương pháp hiệu quả để phát hiện cộng đồng trong đồ thị, tập trung vào tối

ưu hóa hàm modularity. Nó hoạt động bằng cách liên tục gộp các đỉnh vào cộng đồng lân cận để tối đa hóa hàm modularity. Sau đó, các cộng đồng được hợp nhất thành các đỉnh mới, và quá trình lặp lại cho đến khi không thể cải thiện modularity nữa.

1.4. Hàm modularity đánh giá chất lượng cộng đồng mạng

Modularity là một thước đo thường được sử dụng trong lý thuyết đồ thị để đánh giá chất lượng của việc phân chia các đỉnh thành các cộng đồng. Hàm modularity đầu tiên được Newman giới thiệu [9] so sánh số cạnh thực sự tồn tại trong các cộng đồng với số cạnh mong đợi nếu các cạnh được phân bố ngẫu nhiên. Giá trị modularity cao cho thấy rằng có nhiều cạnh hơn trong các cộng đồng so với kỳ vọng, điều này ngụ ý rằng cộng đồng có cấu trúc rõ ràng và hợp lý.

Modularity trong đồ thị vô hướng

Trong đồ thị vô hướng, các cạnh không có hướng, và modularity được tính dựa trên số lượng cạnh giữa các đỉnh trong cùng cộng đồng. Công thức tính modularity cho đồ thị vô hướng được định nghĩa như sau:

$$Q_u = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta(g_i, g_j), \tag{1.4.1}$$

Trong đó:

- A_{ij} là số cạnh giữa đỉnh i và j.
- d_i và d_j lần lượt là bậc của đỉnh i và j.
- m là tổng số cạnh trong đồ thị.
- $\delta(g_i, g_j) = 1$ nếu i và j thuộc cùng một cộng đồng, ngược lại $\delta(g_i, g_j) = 0$.
- $\frac{d_i d_j}{2m}$ là số cạnh trung bình giữa hai đỉnh i và j theo đồ thị cấu hình [10].

Mô hình đồ thị cấu hình là một mô hình tạo đồ thị ngẫu nhiên dựa trên dãy bậc cho trước với tinh thần ghép các nửa cạnh bất kì lại với nhau từ đó

tính được xác suất một nửa cạnh bất kì kết nối tới đỉnh j là $\frac{d_j}{2m-1}$. Bên cạnh đó, bởi đỉnh i có bậc là d_i, do đó số cạnh trung bình xuất hiện giữa hai đỉnh i, j theo mô hình này là:

$$d_i \times \frac{d_j}{2m-1} = \frac{d_i d_j}{2m-1}, \tag{1.4.2}$$

trong đồ thị lớn thì bởi $2m >> 1$, do đó có thể coi số cạnh trung bình là $\frac{d_i d_j}{2m}$.

Công thức (1.4.1) biểu diễn Q_u đo lường mức độ tương thích giữa sự phân chia cộng đồng và số cạnh thực sự trong các cộng đồng. Nếu số cạnh bên trong cộng đồng nhiều hơn kỳ vọng, thì giá trị Q sẽ lớn hơn 0, cho thấy sự phân chia cộng đồng tốt.

Công thức Q_u trên tương đương với biểu diễn sau (xem trong [11]) với cách phân cụm $\mathcal{P} = \{C_1, C_2, ..., C_K\}$:

$$Q_u = \sum_{k=1}^{K} Q_u^{C_k} = \sum_{k=1}^{K} \left[\frac{|E_{C_k}^{in}|}{m} - \left(\frac{D(C_k)}{2m} \right)^2 \right] \tag{1.4.3}$$

với kí hiệu $|E_{C_k}^{in}|$ là số cạnh bên trong cộng đồng C_k, và $D(C_k) = \sum_{i \in C_k} d(i)$ là tổng bậc của các đỉnh trong cộng đồng C_k. Công thức này thể hiện độ đóng góp của từng cộng đồng C_k ($Q_u^{C_k}$) vào giá trị modularity tổng thể. Nếu một cộng đồng có cấu trúc rõ ràng, nó sẽ làm tăng modularity tổng Q_u. Ngược lại, nếu cộng đồng không có cấu trúc rõ ràng, giá trị đóng góp có thể thấp hoặc âm, làm giảm modularity tổng hoặc tăng không đáng kể.

Modularity trong đồ thị có hướng

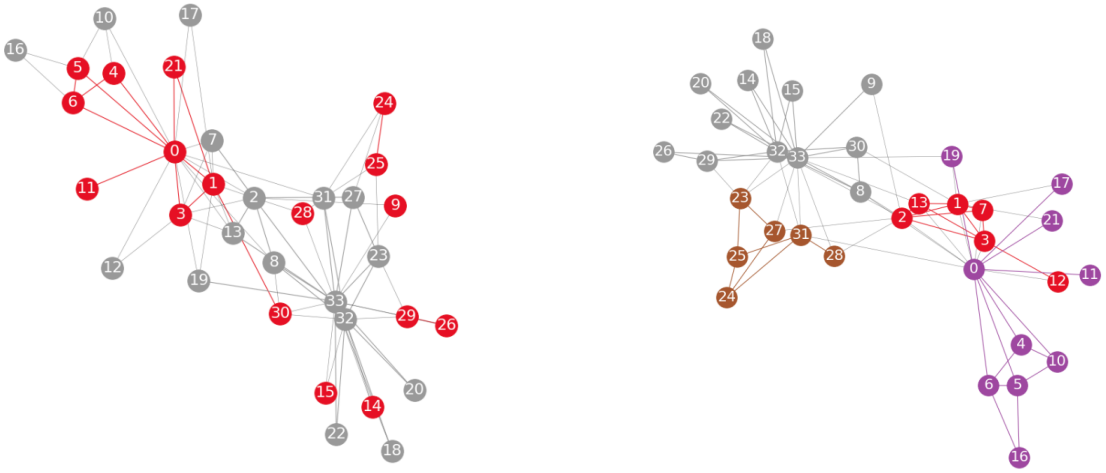
Trong đồ thị có hướng, các cạnh có hướng từ một đỉnh này sang đỉnh khác, và việc tính modularity phức tạp hơn vì cần xét đến hướng của các cạnh. Công thức modularity cho đồ thị có hướng [12] được biểu diễn như sau:

$$Q_d = \frac{1}{m} \sum_{i,j} \left(A_{ij} - \frac{d_i^{\text{out}} d_j^{\text{in}}}{m} \right) \delta(g_i, g_j) \tag{1.4.4}$$

Trong đó:

- A_{ij} là số cạnh có hướng từ đỉnh i đến đỉnh j.
- d_i^{out} là bậc đi ra của đỉnh i

- d_j^{in} là bậc đi vào của đỉnh j.
- m là tổng số cạnh trong đồ thị có hướng.
- $\delta(g_i, g_j) = 1$ nếu i và j thuộc cùng một cộng đồng, ngược lại $\delta(g_i, g_j) = 0$.
- $\frac{d_i^{\text{out}} d_j^{\text{in}}}{m}$ là số cạnh trung bình từ i đến j dựa trên đồ thị cấu hình ngẫu nhiên.



(a) Modularity ≈ -0.04 (b) Modularity ≈ 0.4

Hình 1.1: Minh hoạ giá trị modularity trên đồ thị KarateClub.

Hình 1.1 minh hoạ hai cách phân chia cộng đồng trên đồ thị Karate Club [13], với hai giá trị modularity khác nhau.

- Trong hình 1.1a, đồ thị được chia thành hai loại đỉnh đỏ và xám, trong đó các đỉnh này khá lộn xộn, xen kẽ không thể hiện rõ cấu trúc cộng đồng, dẫn đến modularity thấp. Cụ thể, cộng đồng màu đỏ C có giá trị đóng góp là $Q_u^C \approx -0.02$ khi số lượng cạnh thực tế giữa các đỉnh (= 14) trong cộng đồng nhỏ hơn số lượng cạnh trung bình (= 15.7), đối với cộng đồng màu xám C' cũng đem lại giá trị $Q_u^{C'} \approx -0.02$ khi số lượng cạnh thực tế (= 22) nhỏ hơn số lượng kỳ vọng (= 23.7). Giá trị Q_u tương ứng trường hợp này là xấp xỉ -0.04.
- Trong hình 1.1b, đồ thị được chia thành 4 cộng đồng rõ ràng: tím (C_1), đỏ (C_2), nâu (C_3) và xám (C_4), các đỉnh trong cùng cộng đồng liên kết

chặt chẽ với nhau, dẫn đến giá trị modularity cao hơn. Cụ thể, mức độ đóng góp của các cộng đồng $Q_u^{C_k}$ đều lớn hơn 0 ($Q_u^{C_1} = 0.11, Q_u^{C_2} = 0.08, Q_u^{C_3} = 0.07, Q_u^{C_4} = 0.14$) thể hiện số lượng cạnh thực tế trong mỗi cộng đồng đều lớn hơn số lượng cạnh trung bình. Giá trị modularity của cách phân chia này là 0.4.

Một số hàm modularity khác

Trong các bài báo [14–16], các tác giả trình bày các hàm modularity khác dựa trên những định nghĩa mới về một cộng đồng tốt: ví dụ trong bài báo [14], tác giả còn quan tâm tới mối quan hệ của các cặp đỉnh "không liên quan" mà được phân vào cùng cụm, trong khi tại bài báo [15], tác giả định nghĩa mối quá hệ tốt giữa một cặp đỉnh dựa trên độ dài đường đi giữa các đỉnh và tại bài báo [16] thì đề xuất tính modularity dựa trên độ tương đồng bằng cách thay thế số cạnh giữa các đỉnh thành độ tương đồng giữa 2 đỉnh, một khái niệm tổng quát và có thể định nghĩa bởi nhiều đại lượng khác.

1.5. Phương pháp nhân tử Lagrange

Phương pháp nhân tử Lagrange là một kỹ thuật trong toán học được sử dụng để tìm cực trị (cực đại hoặc cực tiểu) của một hàm số khi có ràng buộc. Phương pháp này đặc biệt hữu ích trong các bài toán tối ưu phi tuyến, nơi mà hàm số cần tối ưu hóa và các ràng buộc không phải là hàm tuyến tính.

Các định nghĩa cơ bản về hàm afin, tập nón, hàm lồi được trình bày rất rõ trong tài liệu [17]. Dưới đây là những định nghĩa, định lý quan trọng về bài toán tối ưu phi tuyến mà luận văn sử dụng.

Định nghĩa 1.5.1 (Bài toán tối ưu phi tuyến [17])**.** *Cho một bài toán tối ưu phi tuyến có dạng:*

$$\min f(x) \quad \textit{với điều kiện} \quad x \in X, \tag{P$^{\text{rb}}$}$$

trong đó: $X = \{x \in \mathbb{R}^N \mid g_i(x) \leq 0, i = 1, \cdots, M, h_j(x) = 0, j = 1, \cdots, J\}$

Cho $x^0 \in X$ là một nghiệm chấp nhận được của bài toán (P_{rb}). Đặt

$$I(x^0) := \{i \in \{1, \ldots, M\} \mid g_i(x^0) = 0\}$$

là tập các chỉ số của ràng buộc $g_i(x) \leq 0$, $i = 1, \ldots, M$, thỏa mãn chặt tại x^0.

Ký hiệu $S(x^0)$ là tập hợp các vectơ v thỏa mãn hệ tuyến tính sau:

$$\begin{cases} \langle \nabla h_j(x^0), v \rangle = 0, & j = 1, \ldots, J \\ \langle \nabla g_i(x^0), v \rangle \leq 0, & i \in I(x^0). \end{cases}$$

Định nghĩa 1.5.2 (Điều kiện chính quy [17]). *Ta nói điều kiện chính quy được thỏa mãn tại x^0 nếu tập nón tiếp xúc với X tại x^0 ($T(X, x^0)$) là tập $S(x^0)$:*

$$T(X, x^0) = S(x^0)$$

Định lí 1.5.3 (Định lý về điều kiện chính quy trong bài toán (P^{rb}) [17]). *Điều kiện chính quy được thỏa mãn tại $x^0 \in X$ nếu có một trong các điều kiện sau:*

1. *Các hàm $h_j(.)$, $(j = 1, \cdots, J)$ và $g_i(.)$, $(i = 1, \cdots, M)$ đều là các hàm afin;*
2. *Các hàm $h_j(.)$, $(j = 1, \cdots, J)$ là afin; các hàm $g_i(.)$, $(i = 1, \cdots, M)$ là lồi và điều kiện Slater sau đây thỏa mãn*

$$\exists \bar{x} \in \mathbb{R}^N : \quad g_i(\bar{x}) < 0,\ i = 1, \cdots, M \quad và \quad h_j(\bar{x}) = 0,\ j = 1, \cdots, J.$$

3. *Các vectơ $\nabla g_i(x^0)$, $i \in I(x^0)$ và $\nabla h_j(x^0)$, $j = 1, \cdots, J$ là độc lập tuyến tính.*

Định nghĩa 1.5.4 (Hàm Lagrange và nhân tử Lagrange [17]). *Hàm Lagrange ứng với bài toán (P^{rb}) được xây dựng như sau:*

$$\mathcal{L}(x, \lambda_1, \cdots, \lambda_M, \mu_1, \cdots, \mu_J) = f(x) + \sum_{i=1}^{M} \lambda_i g_i(x) + \sum_{j=1}^{J} \mu_j h_j(x),$$

trong đó:

- *$\lambda_i \geq 0$ là các nhân tử Lagrange tương ứng với ràng buộc bất đẳng thức $g_i(x)$.*
- *μ_j là các nhân tử Lagrange tương ứng với ràng buộc đẳng thức $h_j(x)$.*

Định lí 1.5.5 (Định lý tồn tại nghiệm cực tiểu địa phương của bài toán (P^{rb}) [17]). *Cho các hàm f, g_i, $i = 1, \cdots, M$ và h_j, $j = 1, \cdots, J$ là các hàm khả vi liên tục trên $\mathbb{R}^N$. Giả sử điểm $x^* \in \mathbb{R}^M$ là nghiệm cực tiểu địa phương của bài toán (P^{rb}) và tại đó điều kiện chính quy được thỏa mãn. Khi đó điều kiện KKT sau là đúng:*

1. $g_i(x^*) \leq 0$, $i = 1, \cdots, M \quad$ *và* $\quad h_j(x^*) = 0$, $j = 1, \cdots, J$;
2. *Tồn tại các số thực* $\lambda_i \geq 0$, $i = 1, \cdots, M$ *và* μ_j, $j = 1, \cdots, J$ *sao cho*

$$\nabla_x \mathcal{L}(x^*, \lambda_1, \cdots, \lambda_M, \mu_1, \cdots, \mu_J) = 0$$

3. $\lambda_i g_i(x^*) = 0$, $i = 1, \cdots, M$.

1.6. Các mô hình sinh đồ thị ngẫu nhiên

Nhãn cộng đồng của một đồ thị xác định mỗi đỉnh thuộc về cộng đồng nào trong đồ thị. Để đánh giá toàn diện và chính xác chất lượng phân cụm, cần biết trước các nhãn của đồ thị có cấu trúc cộng đồng. Điều này giúp so sánh kết quả phân cụm với nhãn đã xác định để đánh giá hiệu quả thuật toán. Do đó, chúng tôi sử dụng một số mô hình sinh đồ thị ngẫu nhiên nhằm tạo ra các đồ thị có cấu trúc cộng đồng rõ ràng và biết trước nhãn của đồ thị, hỗ trợ việc kiểm tra và đánh giá kết quả.

Mô hình phân vùng $l-$vùng [7] Mô hình này có các tham số chính bao gồm: số lượng cộng đồng l, số lượng đỉnh trong mỗi cộng đồng g, xác suất xuất hiện cạnh giữa các đỉnh cùng cộng đồng là p_{in} và xác suất xuất hiện cạnh giữa các đỉnh khác cộng đồng là p_{out}. Với những tham số này, một đồ thị có cấu trúc cộng đồng có thể được tạo ra với các đặc điểm sau:

- Bậc trung bình của một đỉnh: $\mathbb{E}[k] = p_{in}(g-1) + p_{out}g(l-1)$.
- Số lượng đỉnh trong mỗi cụm là như nhau.
- Mọi đỉnh đều có số bậc xấp xỉ nhau.

Mô hình phân vùng ngẫu nhiên Gaussian [7] Khác với mô hình phân vùng $l-$vùng, mô hình này giả định số lượng đỉnh trong mỗi cụm tuân theo phân phối Gaussian. Các tham số chính của mô hình bao gồm:

- Tổng số lượng đỉnh của đồ thị: n.

- Các tham số đặc trưng cho số lượng đỉnh trong mỗi cộng đồng: trung bình μ và phương sai σ.
- Xác suất cạnh trong và ngoài cụm: p_{in} và p_{out}.

Chương 2

Các thuật toán tìm kiếm cộng đồng mạng sử dụng phương pháp tối ưu modularity cục bộ

Phương pháp tối ưu hóa modularity cục bộ là kỹ thuật dùng để tìm kiếm các cộng đồng trong đồ thị sao cho các kết nối bên trong mỗi cộng đồng chặt chẽ hơn so với các kết nối giữa các cộng đồng khác. Thuật toán này hoạt động qua nhiều vòng lặp, mỗi vòng lặp sẽ dần cải thiện việc phân chia các nhóm dựa trên quy tắc đã định sẵn, nhằm tối ưu hóa giá trị modularity, tức là tăng cường sự liên kết nội bộ trong từng cộng đồng.

Một thuật toán nổi bật trong lớp này là thuật toán Louvain [1]. Thuật toán Louvain là một phương pháp nổi bật để phát hiện cộng đồng trong các đồ thị lớn nhờ khả năng tối ưu hóa hiệu quả. Ngoài ra, có những phiên bản cải thiện và mở rộng của thuật toán Louvain như trong các bài báo [2, 18, 19].

2.1. Thuật toán Louvain

Thuật toán Louvain được xây dựng với mục tiêu tối ưu hóa hàm modularity Q_u cho đồ thị vô hướng $G = (V, E)$. Trước khi đi sâu vào chi tiết của thuật toán, chúng ta cùng nhắc lại định nghĩa hàm modularity Q_u [9] như sau:

$$Q_u = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta \left(g_i, g_j \right) \tag{2.1.1}$$

Trong phần này, ta sẽ sử dụng công thức Q_u phiên bản (1.4.3) với cách phân cụm $\mathcal{P} = \{C_1, C_2, ..., C_K\}$ để thấy được rõ độ đóng góp của từng cộng

đồng C_k :

$$Q_u = \sum_{k=1}^{K} Q_u^{C_k} = \sum_{k=1}^{K} \left[\frac{|E_{C_k}^{in}|}{m} - \left(\frac{D(C_k)}{2m} \right)^2 \right] \tag{2.1.2}$$

với kí hiệu $|E_{C_k}^{in}|$ là số cạnh bên trong cộng đồng C_k, và $D(C_k) = \sum_{i \in C_k} d(i)$ là tổng bậc của các đỉnh trong cộng đồng C_k.

Chúng ta có thể mô tả các bước của thuật toán Louvain như sau: ban đầu, mỗi đỉnh trong đồ thị được coi là một cộng đồng riêng lẻ. Thuật toán Louvain sau đó thực hiện qua hai giai đoạn chính và liên tục lặp lại cho đến khi đạt được phân cụm tối ưu.

1. **Giai đoạn 1: Tối ưu hóa modularity bằng cách di chuyển các đỉnh**
   Trong giai đoạn này, mỗi đỉnh $i \in V$ được xem xét để di chuyển vào một trong các cộng đồng kề với nó. Mỗi lần thử di chuyển đỉnh i tới cụm C, chúng ta sẽ tính sự thay đổi modularity, ký hiệu là $\Delta Q_u^{i \to C}$. Đỉnh i sẽ được chuyển đến cộng đồng C nào đó mà tại đó $\Delta Q_u^{i \to C}$ đạt giá trị lớn nhất. Quá trình này lặp lại cho đến khi tất cả các đỉnh đã được xét và không còn sự cải thiện nào thêm về giá trị modularity.

   Chúng ta tính toán sự thay đổi Q_u khi di chuyển đỉnh i từ cộng đồng C' sang cộng đồng C bằng cách so sánh giá trị modularity trước và sau khi đỉnh i được di chuyển từ cộng đồng C' sang C. Giá trị thay đổi modularity này được ký hiệu là $\Delta Q_u^{i \to C}$ và được tính cụ thể như sau:

$$\begin{aligned}
\Delta Q_u^{i \to C} &= \left(Q_u^{C \cup \{i\}} + Q_u^{C' \setminus \{i\}} \right) - \left(Q_u^{C} + Q_u^{C'} \right) \\
&= \left[\frac{|E_C^{in}| + d_i^C}{m} - \left(\frac{D(C) + d_i}{2m} \right)^2 + \frac{|E_{C'}^{in}| - d_i^{C'}}{m} - \left(\frac{D(C') - d_i}{2m} \right)^2 \right] \\
&\quad - \left[\frac{|E_C^{in}|}{m} - \left(\frac{D(C)}{2m} \right)^2 + \frac{|E_{C'}^{in}|}{m} - \left(\frac{D(C')}{2m} \right)^2 \right]
\end{aligned} \tag{2.1.3}$$

   với $|E_C^{in}|, |E_{C'}^{in}|$ lần lượt là số cạnh bên trong cộng đồng C, C'; $D(C), D(C')$ lần lượt là tổng bậc của các đỉnh trong cộng đồng C, C'; d_i là tổng bậc đỉnh i, và $d_i^C, d_i^{C'}$ lần lượt là tổng số cạnh đỉnh i liên kết với các đỉnh khác trong cộng đồng C, C'.

Đỉnh i sẽ được di chuyển vào cộng đồng C_{max} sao cho giá trị $\Delta Q_u^{i \to C}$ lớn nhất:

$$C_{max} = \text{argmax}_C \Delta Q_u^{i \to C} \tag{2.1.4}$$

Nếu không có cộng đồng nào cải thiện được modularity, đỉnh i sẽ ở lại cộng đồng ban đầu. Giai đoạn này tiếp tục cho đến khi không thể cải thiện thêm modularity. Bởi vì giá trị hàm modularity $Q_u \leq 1$ nên giai đoạn này sẽ dừng lại được.

2. **Giai đoạn 2: Tổng hợp cộng đồng và tạo đồ thị mới**
   Sau khi kết thúc giai đoạn 1, các cộng đồng được tổng hợp lại thành các siêu đỉnh để tạo thành một đồ thị mới. Trọng số giữa các siêu đỉnh được tính bằng tổng trọng số các cạnh giữa các cộng đồng tương ứng trong đồ thị ban đầu. Tức là, trọng số giữa hai siêu đỉnh C_x và C_y được tính bằng:

$$A_{C_x C_y} = \sum_{i \in C_x, j \in C_y} A_{ij} \tag{2.1.5}$$

   với A_{ij} là trọng số cạnh giữa hai đỉnh i và j trong đồ thị ban đầu.

   Sau đó, giai đoạn 1 được lặp lại trên đồ thị mới này. Quá trình này tiếp tục cho đến khi giá trị modularity không còn cải thiện thêm.

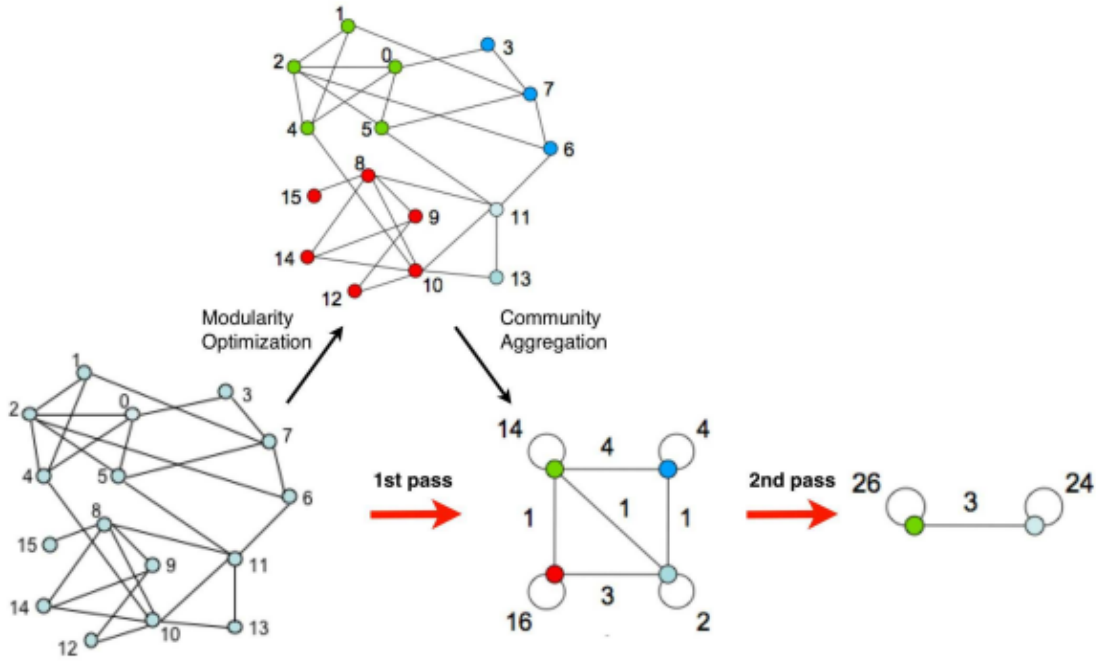
Hình 2.1 (nguồn: bài báo [1]) minh hoạ thuật toán Louvain qua hai cấp độ. Mã giả của thuật toán Louvain được thể hiện trong Thuật toán 1 sau:

Algorithm 1 Thuật toán Louvain

Input: Đồ thị $G = (V, E)$
Output: Các cộng đồng $C_1, C_2, \ldots, C_k$

Biến chỉ cấp đang xét: $l = 0$ và khởi tạo $\mathcal{P} = \emptyset$
Đồ thị cấp 0: $G^0 = (V^0, E^0) = (V, E) = G$
while ($l = 0$ *hoặc* $|\mathcal{P}| = |V^l|$) **do**
    /* Xét đồ thị tại cấp l */
    Khởi tạo mỗi đỉnh là một cộng đồng riêng lẻ: $\mathcal{P} = \{\{1\}, \{2\}, ..., \{|V^l|\}\}$
    /* Giai đoạn 1: Tối ưu modularity */
    Biến chỉ số vòng lặp: $t = 0$
    Đặt $Q(\mathcal{P})$ là giá trị modularity của phân cụm $\mathcal{P}$.
    Gán $Q_{old} = Q(\mathcal{P})$
    while ($Q(\mathcal{P}) > Q_{old}$ *hoặc* $t = 0$) **do**
        $t = t + 1$
        Gán $Q_{old} = Q(\mathcal{P})$
        for $i \in \{1, 2, ..., |V^l|\}$ **do**
            /* Tính độ cải thiện modularity khi $i \to C$ */
            Tập hợp các cộng đồng lân cận: $T = \{C | C \in \mathcal{P}, (i, j) \in E, j \in C\}$
            for $C \in T$ **do**
                Tính $\Delta Q_u^{i \to C}$ bằng công thức (2.1.3)
            end
            if $\max \Delta Q_u^{i \to C} > 0$ **then**
                Đặt $C_{max} = \text{argmax}_C \Delta Q_u^{i \to C}$
                Cập nhật $\mathcal{P}$: Di chuyển đỉnh i từ cộng đồng hiện tại vào cộng đồng C_{max}
            end
        end
        Tính lại giá trị $Q(\mathcal{P})$
    end
    /* Giai đoạn 2: Tổng hợp cộng đồng, tạo đồ thị mới */
    Đặt $\mathcal{V} = \{(C, D) | C, D \in \mathcal{P}\}$
    for $(C_x, C_y) \in \mathcal{V}$ **do**
        Tính trọng số cạnh $A_{C_x C_y}$ giữa các siêu đỉnh C_x, C_y trong đồ thị mới G^{l+1} theo công thức (2.1.5)
    end
    $l = l + 1$
end
return Cách phân cụm $\mathcal{P}$ với các đỉnh của đồ thị ban đầu (G)

Trong bài báo [20], hai tác giả Lancichinetti và Fortunato đã chỉ ra rằng về cơ bản, độ phức tạp của thuật toán Louvain là tuyến tính theo số cạnh của đồ



Hình 2.1: Minh hoạ từng giai đoạn của thuật toán Louvain khi áp dụng lên một đồ thị nhỏ. Tại giai đoạn 1 của cấp đầu tiên, bốn cộng đồng (đỏ, xanh lam, xanh lục và xanh nhạt) được phát hiện. Ở giai đoạn 2, các cộng đồng này trở thành bốn siêu đỉnh, và một đồ thị mới được tạo ra với trọng số mới. Quá trình này tiếp tục lặp lại ở các cấp độ tiếp theo.

thị. Do đó có thể đánh giá độ phức tạp của thuật toán Louvain là $O(m)$.

Ngoài ra, thuật toán Louvain cũng được mở rộng cho trường hợp có hướng, các tác giả của bài báo [19] đã mở rộng thuật toán Louvain cho đồ thị có hướng bằng cách thay thế hàm Q_u (trong công thức (1.4.1)) bằng hàm Q_d cho đồ thị có hướng:

$$Q_d = \frac{1}{m}\sum_{i,j}\left[A_{ij} - \frac{d_i^{out} d_j^{in}}{m}\right]\delta(g_i, g_j), \tag{2.1.6}$$

2.2. Thuật toán Leiden

Đã có nhiều nghiên cứu mở rộng, cải thiện thuật toán Louvain như trong các bài báo [2, 18, 19]. Trong đó, thuật toán Leiden [2] được nhiều người quan tâm, vì nó không chỉ cải thiện tính hiệu quả mà còn giảm thời gian tính toán. Trong phần này, chúng tôi sẽ trình bày chi tiết về thuật toán Leiden.

Trong bài báo của Traag và cộng sự (2019) [2], các tác giả đã chỉ ra rằng thuật toán Louvain có thể gặp vấn đề với việc duy trì kết nối giữa các cụm con bên trong một cụm chính. Cụ thể, các cụm con này có thể không còn liên kết với nhau sau một số lần lặp của quá trình tối ưu modularity. Ví dụ, điều này có thể xảy ra khi một đỉnh trung gian, đóng vai trò cầu nối giữa hai cụm con, bị di chuyển sang cộng đồng khác do có nhiều kết nối bên ngoài hơn so với kết nối bên trong cộng đồng hiện tại của nó. Khi đỉnh này rời đi, các cụm con sẽ bị tách ra, dẫn đến việc mất kết nối trong cộng đồng.

Thuật toán Leiden ra đời để khắc phục vấn đề này. Mặc dù thuật toán Louvain có thể tối ưu modularity, nhưng nó không đảm bảo tính kết nối bên trong các cộng đồng. Thuật toán Leiden thêm một giai đoạn tinh chỉnh sau quá trình phân cụm ban đầu, nhằm đảm bảo rằng các cộng đồng sau khi được tối ưu modularity vẫn giữ được tính liên kết nội bộ.

Một số định nghĩa về kích thước của tập đỉnh và điều kiện liên kết chặt chẽ trong thuật toán Leiden được trình bày như sau:

Định nghĩa 2.2.1 (Kích thước của tập hợp đỉnh [2])**.** *Cho tập hợp $S = \{s_1, s_2, ..., s_k\}$ với $s_i (i = 1, 2, ..., k)$ là các đỉnh thuộc đồ thị gốc ban đầu. Khi đó kích thước của tập S được xác định như sau:*

$$\|S\| = \sum_{i=1}^{k} d_{s_i}, \tag{2.2.1}$$

trong đó d_{s_i} là bậc của đỉnh s_i.

Định nghĩa 2.2.2 (Điều kiện liên kết chặt chẽ [2])**.** *Hai tập R và S thuộc cộng đồng $C \in \mathcal{P}$ được gọi là liên kết chặt chẽ nếu:*

$$E(R, S) \geq \frac{\gamma}{2m} \|R\| \|S\|, \tag{2.2.2}$$

với $E(R, S)$ là tổng số cạnh nối giữa 2 tập R, S, $\|R\|, \|S\|$ lần lượt là kích thước của hai tập R, S; γ là tham số độ phân giải được xác định trong các công thức (2.2.3), (2.2.4).

Thuật toán Leiden [2] dành cho đồ thị vô hướng sử dụng một hàm modularity mở rộng, dựa trên hàm Q_u. Hàm modularity mở rộng này, được ký hiệu là Q_f, có thể được coi như một dạng tổng quát của hàm Q_u.

$$Q_f = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \gamma \frac{d_i d_j}{2m} \right] \delta(g_i, g_j), \tag{2.2.3}$$

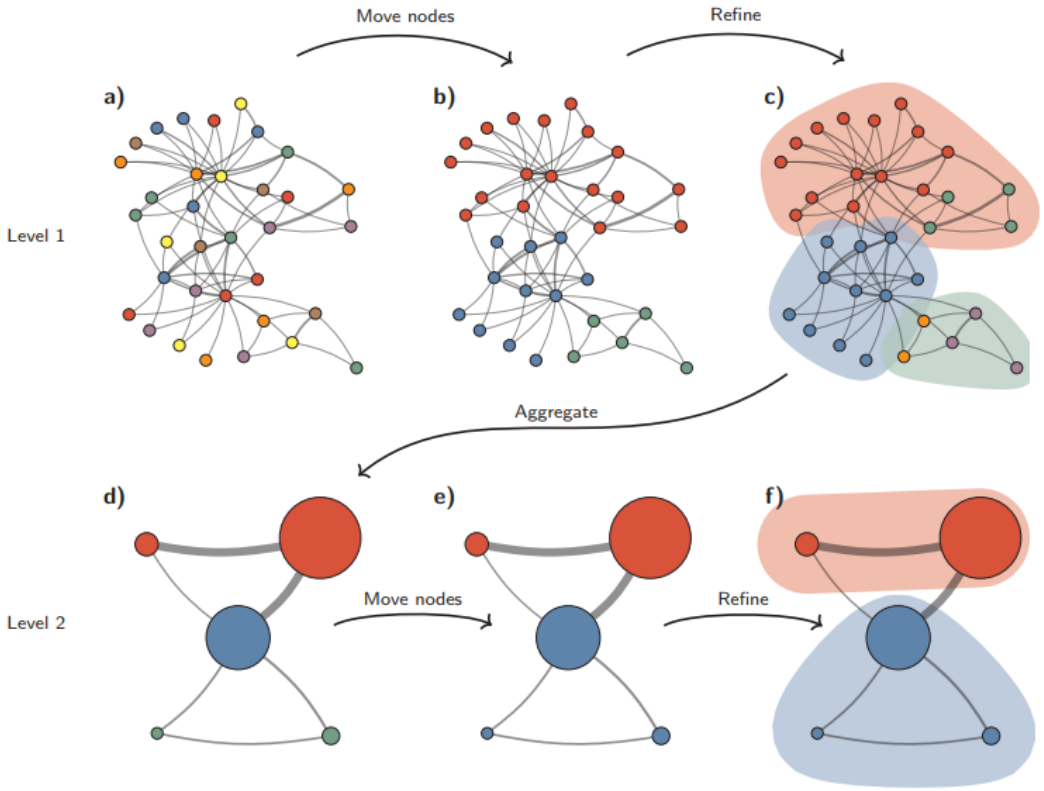
với γ là hệ số phân giải, độ phân giải cao dẫn tới nhiều cộng đồng hơn trong khi độ phân giải thấp thì số lượng cộng đồng sẽ ít đi. Giá trị hàm Q_f của cách phân cụm $\mathcal{P} = \{C_1, C_2, ..., C_K\}$ cũng có thể biểu diễn công thức như sau với $Q_f^{C_k}$ là độ đóng góp của cộng đồng C_k:

$$Q_f = \sum_{k=1}^{K} Q_f^{C_k} = \sum_{k=1}^{K} \left[\frac{|E_{C_k}^{in}|}{m} - \gamma \left(\frac{D(C_k)}{2m} \right)^2 \right] \tag{2.2.4}$$

với kí hiệu $|E_{C_k}^{in}|$ là số cạnh bên trong cộng đồng C_k, và $D(C_k) = \sum_{i \in C_k} d(i)$ là tổng bậc của các đỉnh trong cộng đồng C_k. Công thức này thể hiện độ đóng góp của từng cộng đồng C_k ($Q_f^{C_k}$) vào giá trị modularity tổng thể.

Cấu trúc của thuật toán Leiden bao gồm ba giai đoạn chính. Giai đoạn 1 là di chuyển các đỉnh để tối ưu hóa modularity, trong đó có cải tiến so với thuật toán Louvain bằng cách sử dụng hàng đợi nhằm giảm số đỉnh cần xét, từ đó thu được phân cụm $\mathcal{P}$. Giai đoạn 2 là tinh chỉnh các cụm đã xuất hiện trong giai đoạn 1, nhằm xác định các cụm con trong mỗi cộng đồng và thu được phân cụm mới $\mathcal{P}_{refine}$. Cuối cùng, giai đoạn 3 thực hiện việc tổng hợp các cộng đồng, tạo đồ thị mới bằng cách sử dụng hai kết quả phân cụm $\mathcal{P}, \mathcal{P}_{refine}$.

Hình 2.2 (nguồn: bài báo [2]) minh họa quá trình thực hiện thuật toán Leiden qua hai cấp liên tiếp. Tại cấp 1 (level 1), trong giai đoạn 1 tương ứng với hình b, thuật toán đã phát hiện ba cộng đồng: đỏ, xanh lục và xanh lam, tương ứng với cách chia $\mathcal{P}$. Trong hình c, giai đoạn tinh chỉnh được thực hiện, khi này có sự thay đổi từ $\mathcal{P}$ sang $\mathcal{P}_{refine}$. Cụ thể, cộng đồng đỏ của $\mathcal{P}$ được chia thành hai cộng đồng con: màu đỏ và màu xanh tươi, trong khi đó cộng đồng xanh lam giữ nguyên, và cộng đồng xanh lục ban đầu cũng được chia thành hai cộng đồng nhỏ hơn: màu da cam và màu tím. Hình d thể hiện kết quả của giai đoạn tổng hợp. Các cộng đồng nhỏ trong $\mathcal{P}_{refine}$ tương ứng với



Hình 2.2: Minh họa cấu trúc ba giai đoạn của thuật toán Leiden.

một đỉnh trong đồ thị mới, nhưng vẫn được xét là thuộc cùng cụm dựa trên kết quả từ $\mathcal{P}$. Các hình e và f thể hiện vòng lặp tiếp theo của đồ thị tổng hợp.

Chi tiết thuật toán Leiden được giải thích như sau:

Giai đoạn tối ưu modularity: Đầu tiên, chúng ta tạo ra một hàng đợi $\mathcal{K}$ bao gồm tất cả các đỉnh của đồ thị. Sau đó, lần lượt xét từng đỉnh $v \in \mathcal{K}$. Mỗi khi một đỉnh v được xét xong, nó sẽ bị loại khỏi hàng đợi, giả sử cộng đồng hiện tại của đỉnh v là C. Tiếp theo, đỉnh v sẽ được xem xét để di chuyển sang các cộng đồng lân cận nhằm tìm ra cộng đồng C' mang lại mức cải thiện giá trị modularity lớn nhất. Công thức $\Delta Q_f^{v \to C'}$ được xác định như sau:

$$\Delta Q_f^{v \to C'} = \left(Q_f^{C' \cup \{v\}} + Q_f^{C \setminus \{v\}}\right) - \left(Q_f^{C} + Q_f^{C'}\right)$$

$$= \left[\frac{|E_{C'}^{in}| + d_v^{C'}}{m} - \gamma\left(\frac{D(C') + d_v}{2m}\right)^2 + \frac{|E_C^{in}| - d_v^{C}}{m} - \gamma\left(\frac{D(C) - d_v}{2m}\right)^2\right]$$

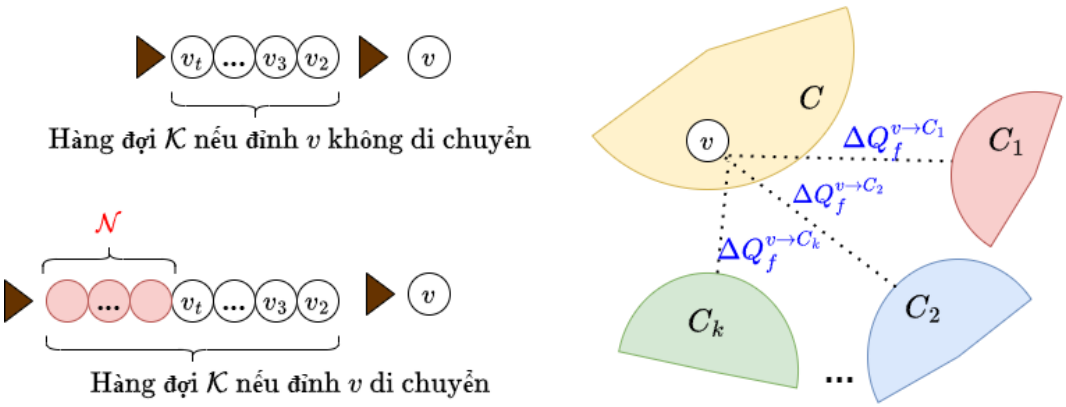
$$- \left[\frac{|E_C^{in}|}{m} - \gamma\left(\frac{D(C)}{2m}\right)^2 + \frac{|E_{C'}^{in}|}{m} - \gamma\left(\frac{D(C')}{2m}\right)^2\right], \tag{2.2.5}$$

Nếu đỉnh v không di chuyển ($\nexists C' : \Delta Q_f^{v \to C'} > 0$) thì hàng đợi $\mathcal{K}$ sẽ loại đi đỉnh v. Còn trong trường hợp đỉnh v chuyển đến cộng đồng C' nào đó, điều này có thể ảnh hưởng tới các đỉnh kề của v, khiến các đỉnh này có thể cần được xét tới. Do đó, một tập hợp các đỉnh kề với v mà chưa thuộc cộng đồng C' (kí hiệu là tập $\mathcal{N}$) và đã bị loại khỏi hàng đợi $\mathcal{K}$ sẽ được thêm vào hàng đợi $\mathcal{K}$. Cụ thể:

$$\mathcal{N} = \{u \mid (u, v) \in E, u \notin C'\}, \tag{2.2.6}$$

$$\mathcal{K} = \mathcal{K} \cup (\mathcal{N} \setminus \mathcal{K}), \tag{2.2.7}$$

Quá trình xét các đỉnh v như vậy sẽ tiếp tục cho đến khi hàng đợi $\mathcal{K}$ trống hoàn toàn. Hình 2.3 minh hoạ việc sử dụng hàng đợi trong giai đoạn 1 của thuật toán Leiden: hàng đợi $\mathcal{K}$ chỉ có thể tăng thêm số lượng các đỉnh cần xét khi đỉnh v có sự di chuyển vào cộng đồng khác.



Hình 2.3: Minh họa sử dụng hàng đợi trong giai đoạn 1 của thuật toán Leiden.

Giai đoạn tinh chỉnh: Sau khi hoàn thành giai đoạn tối ưu modularity, chúng ta có được một phân cụm ban đầu $\mathcal{P}$. Trong giai đoạn tinh chỉnh, $\mathcal{P}$ có thể được chia nhỏ thành các cụm con, tạo ra một phân cụm mới gọi là $\mathcal{P}_{refine}$. Cụ thể, cách phân cụm tinh chỉnh $\mathcal{P}_{refine}$ được xây dựng bằng cách di chuyển từng

đỉnh trong mỗi cộng đồng $C \in \mathcal{P}$ lại với nhau (dựa trên điều kiện liên kết chặt chẽ).

Tại thời điểm ban đầu, $\mathcal{P}_{refine} = \{\{1\}, \{2\}, ..., \{n\}\}$. Với mỗi cộng đồng $C \in \mathcal{P}$, thuật toán sẽ thực hiện quá trình tinh chỉnh như sau:

- *Xác định danh sách các đỉnh có thể di chuyển:* Bước đầu là xác định tập R bao gồm các đỉnh trong cộng đồng C, mà nó "liên kết chặt chẽ" với phần còn lại của cộng đồng C, cụ thể như sau:

$$R = \{v | v \in C, E(v, C - v) \geq \frac{\gamma}{2m} \|v\| (\|C\| - \|v\|)\}, \tag{2.2.8}$$

  Các đỉnh v sẽ được xem xét di chuyển nếu nó là đỉnh thuộc tập R và thoả mãn ($\{v | \{v\} \in \mathcal{P}_{refine}\}$).

- *Xác định các cộng đồng có thể chuyển đến:* Ta xác định tập các cộng đồng $\mathcal{T}$ mà các đỉnh có thể xem xét chuyển đến là những cộng đồng thuộc $\mathcal{P}_{refine}$ và có "liên kết chặt chẽ" với phần còn lại của cụm C, cụ thể:

$$\mathcal{T} = \{D | D \in \mathcal{P}_{refine}, D \subseteq C, E(D, C - D) \geq \frac{\gamma}{2m} \|D\| (\|C\| - \|D\|)\}, \tag{2.2.9}$$

- *Điều kiện di chuyển:* Tiếp đến thì đỉnh v sẽ di chuyển tới một cộng đồng $D \in \mathcal{T}$ với xác suất $P(D)$ dựa trên độ cải thiện modularity $\Delta Q_f^{i \to D}$ với θ là tham số:

$$P(D) \sim \exp(\frac{1}{\theta} \Delta Q_f^{i \to D}) \text{ nếu } \Delta Q_f^{i \to D} > 0 \text{ và } D \in \mathcal{T} \tag{2.2.10}$$

- Sau khi đỉnh v di chuyển thì ta cập nhật $\mathcal{P}_{refine}$ và tiếp tục lặp lại vòng lặp với các đỉnh $v' \in R$ và $\{v'\} \in \mathcal{P}_{refine}$.

- Phần "tinh chỉnh" cho cộng đồng $C \in \mathcal{P}$ sẽ kết thúc khi không còn đỉnh nào vừa đứng độc lập và thuộc tập R.

Giai đoạn tổng hợp, tạo đồ thị mới từ kết quả $\mathcal{P}$ và $\mathcal{P}_{refine}$: Sau khi nhận được phân cụm tinh chỉnh $\mathcal{P}_{refine}$ từ giai đoạn tinh chỉnh, thuật toán sẽ tổng hợp các cụm con này thành các siêu đỉnh. Thuật toán Leiden khởi tạo cộng đồng cho các siêu đỉnh mới này dựa trên cách phân cụm từ giai đoạn một $\mathcal{P}$.

Thuật toán Leiden có thể biểu diễn qua mã giả của thuật toán 2 dưới đây.

Trong bài báo [2], các tác giả đã chỉ ra với cách xét theo hàng đợi $\mathcal{K}$ ở trên, thuật toán Leiden sẽ nhanh hơn so với thuật toán Louvain vì thuật toán Louvain phải xét toàn bộ đỉnh rất nhiều lần. Bên cạnh đó, dựa trên thực nghiệm trong bài báo [2], và được trình bày trong phần 2.3.1 dưới đây, thuật toán Leiden chạy nhanh hơn nhiều so với thuật toán Louvain.

2.3. Một số thí nghiệm

Trong phần này, chúng tôi sẽ minh hoạ độ hiệu quả của các thuật toán Louvain, Leiden trên các đô thị thực (được thể hiện trong các bài báo gốc [1,2]) và thí nghiệm trên một số đồ thị ngẫu nhiên.

2.3.1. Thí nghiệm trên đồ thị thực

Thí nghiệm trên thuật toán Louvain

Trong phần này, chúng tôi sẽ trình bày kết quả từ bài báo gốc [1] về so sánh tính hiệu quả của thuật toán Louvain với các thuật toán CNM [10], WalkTrap [21], và WT [22], dựa trên hai tiêu chí: chất lượng phân cụm (được đánh giá qua giá trị hàm modularity Q_u) và thời gian chạy. Các thí nghiệm được thực hiện trên một số mạng thực được liệt kê trong Bảng 2.1 dưới đây. Chúng ta sẽ sử dụng các ký hiệu sau: $k \sim$ nghìn, $M \sim$ triệu, $B \sim$ tỉ.

Đồ thị	(V, E)	Mô tả
Karate [23]	$(34, 78)$	Mối liên hệ giữa các thành viên trong CLB Karate
Arxiv [24]	$(9k, 24k)$	Đồ thị gồm 9000 bài báo khoa học và trích dẫn
Web nd.edu [25]	$(325k, 1M)$	Đồ thị trang web gồm vài trăm nghìn trang web (miền nd.edu)
Phone [26]	$(2.6M, 6.3M)$	Mạng lưới điện thoại của công ty điện thoại di động Bỉ.
Web uk-2005 [27]	$(39M, 783M)$	Một đồ thị con của miền .uk
Web WebBase 2001 [28]	$(118M, 1B)$	Đồ thị được thu thập bởi Stanford WebBase

Bảng 2.1: Danh sách các đồ thị thực được sử dụng trong thí nghiệm.

Kết quả thí nghiệm được trình bày trong trong Bảng 2.2 dưới đây (nguồn: bài báo [1]), so sánh hiệu suất của thuật toán Louvain với các thuật toán khác dựa trên hai tiêu chí: giá trị modularity Q_u và thời gian chạy. Mỗi ô trong bảng được biểu diễn dưới dạng $Q_u/T(s)$, trong đó Q_u là giá trị hàm modularity và $T(s)$ là thời gian chạy (tính bằng giây). Ký hiệu $-/-$ cho biết rằng thuật toán không thể thực thi trên đồ thị tương ứng.

	Karate	Arxiv	Web nd.edu	Phone	Web uk-2005	Web WebBase 2001
CNM [10]	0.38/0s	0.77/3.6s	0.93/5034s	-/-	-/-	-/-
WalkTrap [21]	0.42/0s	0.76/3.3s	0.90/6666s	-/-	-/-	-/-
WT [22]	0.42/0s	0.76/0.7s	0.90/248s	0.56/464s	-/-	-/-
Louvain	0.42/0s	0.81/0s	0.94/3s	0.77/134s	0.98/738s	0.98/152p

Bảng 2.2: Kết quả so sánh thuật toán trên các đồ thị khác nhau.

Nhận xét 2.3.1. *Thuật toán Louvain thể hiện sự vượt trội rõ rệt, đặc biệt khi kích thước đồ thị tăng lên, qua các trường hợp sau:*

- ***Đồ thị nhỏ (Karate):***
  Với đồ thị Karate có vài chục đỉnh và cạnh, cả bốn thuật toán đều có thời gian xử lý gần như tức thì (0s được hiểu ở đây là thời gian xử lý rất ngắn), không có sự khác biệt đáng kể. Giá trị modularity Q_u của thuật toán Louvain đem lại bằng với hai thuật toán WalkTrap, WT và lớn hơn thuật toán CNM.

- ***Đồ thị trung bình (ArXiv):***
  Khi xét đồ thị ArXiv với gần 10 nghìn đỉnh và hàng chục nghìn cạnh, thuật toán Louvain vẫn xử lý gần như ngay lập tức. Trong khi đó, các thuật toán khác mất từ 0.7 giây đến hơn 3 giây để hoàn thành. Ngoài ra, đã có sự vượt lên của giá trị Q_u từ thuật toán Louvain (0.81) khi so với giá trị 0.76, 0.77 của các thuật toán còn lại.

- ***Đồ thị rất lớn (Web uk-2005, WebBase):***
  Đối với các mạng lưới có kích thước khổng lồ như Web uk-2005 hay WebBase (1

tỉ cạnh), các thuật toán khác đều không thể xử lý. Tuy nhiên, thuật toán Louvain vẫn xử lý được với tốc độ rất nhanh, khẳng định khả năng vượt trội của nó đối với các đồ thị lớn.

Thí nghiệm trên thuật toán Leiden

Trong phần này, chúng tôi sẽ trình bày kết quả so sánh giữa thuật toán Leiden [2] và thuật toán Louvain được trình bày trong [2], dựa trên các tiêu chí: chất lượng phân cụm (được đánh giá thông qua hàm modularity Q_u), độ kết nối trong cộng đồng và thời gian chạy của thuật toán. Mỗi thí nghiệm đã được chạy 10 lần để thể hiện sự ổn định của kết quả đánh giá.

So sánh chất lượng phân cụm thông qua hàm modularity Q_u

Bảng 2.3 (nguồn: bài báo [2]) gồm các mạng lưới thực được thực hiện với một số kí hiệu đơn vị: $k \sim$ nghìn, $M \sim$ triệu, $B \sim$ tỉ. Trong khi đó, bảng 2.4 thể hiện sự so sánh giá trị Q_u của hai thuật toán Louvain và Leiden sau một số cấp đầu tiên.

Đồ thị	(V, E)	Mô tả
DBLP [29]	$(317k, 2.1M)$	Đồ thị trích dẫn giữa các bài báo trên cơ sở dữ liệu DBLP
Amazon [29]	$(335k, 1.9M)$	Đồ thị sản phẩm trên Amazon
IMDB [30]	$(374k, 30.3M)$	Đồ thị phim trên hệ thống IMDB
Live Journal [29]	$(3.9M, 69.6M)$	Đồ thị tương tác giữa các thành viên trên hệ thống Live Journal
Web of Science	$(9.8M, 208M)$	Đồ thị giữa các trang web khoa học
Web UK [31]	$(39.2M, 1.5B)$	Đồ thị con giữa các trang web có miền .uk

Bảng 2.3: Các mạng thực được sử dụng trong thí nghiệm của bài báo Leiden.

Kết quả thí nghiệm về giá trị Q_u thu được sau 10 cấp (tương ứng là 9 giai đoạn tạo đồ thị mới) thể hiện thuật toán Leiden đem lại giá trị modularity có cao hơn nhưng không đáng kể. Cụ thể, đối với đồ thị lớn như DBLP thì thuật toán Leiden tốt hơn 0.1; đối với đồ thị lớn hơn như IMDB mức độ chênh

lệch chỉ là 0.0007 còn với đồ thị rất lớn như Web UK thì mức độ chênh lệch là 0.0005.

Đồ thị	(V, E)	Q_u Louvain	Q_u Leiden
DBLP	$(317k, 2.1M)$	0.8262	0.8387
Amazon	$(335k, 1.9M)$	0.9301	0.9341
IMDB	$(374k, 30.3M)$	0.7062	0.7069
Live Journal	$(3.9M, 69.6M)$	0.7653	0.7739
Web of Science	$(9.8M, 208M)$	0.7911	0.7951
Web UK	$(39.2M, 1.5B)$	0.9796	0.9801

Bảng 2.4: Giá trị modularity lớn nhất sau 10 cấp đầu tiên trong 10 lần chạy hai thuật toán.

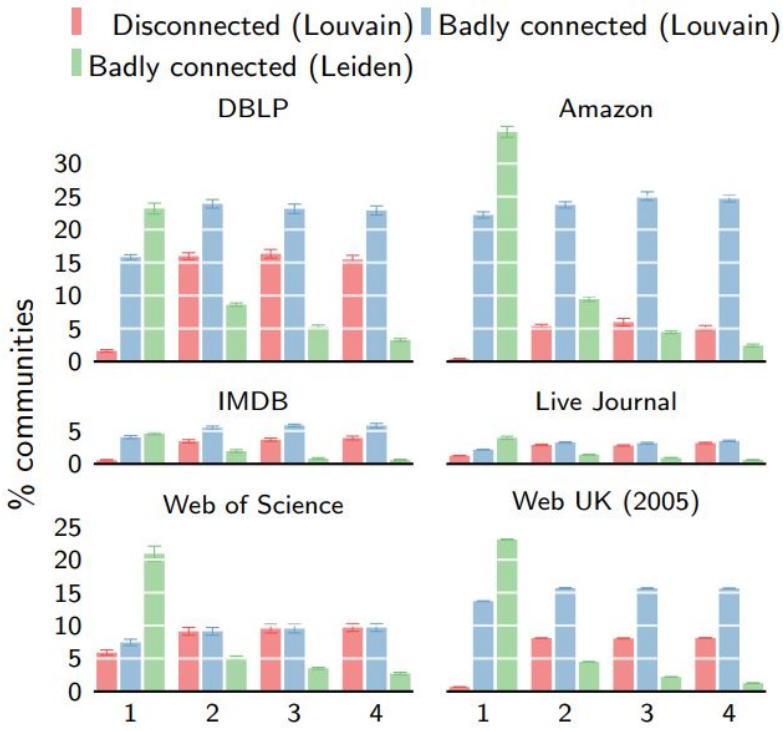
So sánh độ kết nối trong cộng đồng qua từng vòng lặp

Dựa trên Định nghĩa 2.2.2 về điều kiện liên kết chặt chẽ giữa các tập con, một cộng đồng được coi là mất kết nối khi tồn tại hai cộng đồng con R và S không có bất kỳ cạnh kết nối trực tiếp nào giữa chúng. Một cộng đồng được xem là kết nối kém khi tồn tại hai cộng đồng con R, S mà số lượng cạnh kết nối giữa hai cộng đồng con (ký hiệu là $E(R, S)$) là rất ít.

Hình 2.4 so sánh chất lượng kết nối của các cộng đồng được sinh ra từ hai thuật toán Louvain và Leiden qua các vòng lặp. Trong hình, trục hoành biểu thị số thứ tự vòng lặp, trục tung cho thấy phần trăm các cộng đồng bị kết nối yếu (màu xanh dương cho Louvain, màu xanh lục cho Leiden) hoặc mất kết nối (màu đỏ cho Louvain). Mỗi đồ thị thí nghiệm tương ứng với một biểu đồ riêng, kèm theo tên của đồ thị đó được hiển thị ở phía trên.

Nhận xét 2.3.2. *Từ hình 2.4, ta có thể rút ra các nhận xét về hiệu quả của thuật toán Louvain và Leiden như sau:*
Thuật toán Louvain *gặp phải hai vấn đề chính:*

- *Các cộng đồng do Louvain sinh ra có thể bị mất kết nối hoặc kết nối kém.*
- *Phần trăm các cộng đồng bị kết nối yếu không những không giảm qua các vòng lặp mà còn tăng lên. Ví dụ, ở vòng lặp đầu tiên, 10-15% cộng đồng bị kết nối*



Hình 2.4: Minh họa kết quả tỉ lệ số cụm có kết nối yếu qua từng vòng lặp của 2 thuật toán Louvain và Leiden.

kém, và sau vài vòng lặp, con số này có thể tăng lên đến 20-25% đối với đồ thị DBLP và Amazon, hoặc 10-15% với các đồ thị lớn hơn như Web of Science và Web UK.

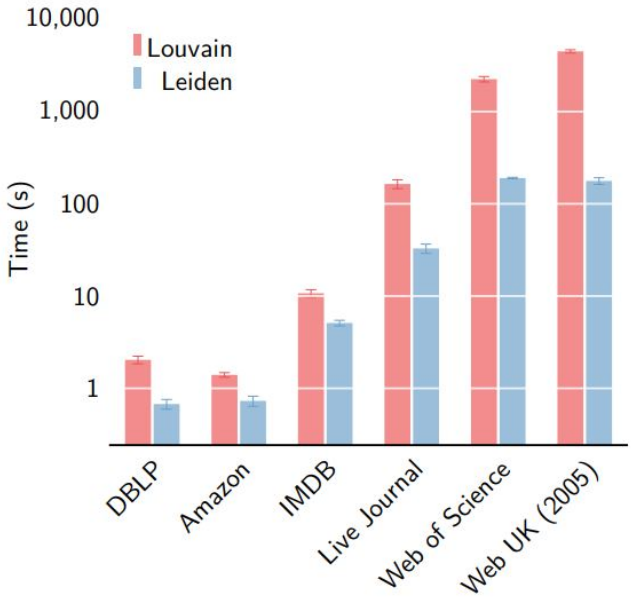
- *Tương tự, tỷ lệ các cộng đồng bị mất kết nối cũng tăng dần, thường dao động từ 5-15% qua các vòng lặp.*

***Thuật toán Leiden**, nhờ có giai đoạn tinh chỉnh, khắc phục được những hạn chế của Louvain:*

- *Không có cộng đồng nào bị mất kết nối trong quá trình xử lý của Leiden.*
- *Dù tỷ lệ cộng đồng bị kết nối yếu ban đầu khá cao (khoảng 20-25%), nhưng sau các vòng lặp, tỷ lệ này giảm đáng kể, xuống dưới 5% sau vòng lặp thứ tư đối với đồ thị DBLP, Amazon, Web of Science và Web UK.*
- *Sau vòng lặp thứ hai, Leiden cho thấy hiệu quả vượt trội khi tỷ lệ cộng đồng bị kết nối kém trở nên rất nhỏ, gần như không đáng kể so với Louvain.*

So sánh thời gian chạy thuật toán

Từ lý thuyết của hai thuật toán Louvain và Leiden, ta biết rằng sau giai đoạn tạo đồ thị mới, số lượng đỉnh sẽ giảm đáng kể. Do đó, thời gian xử lý trong vòng lặp đầu tiên (khi phải xét đến số lượng đỉnh ban đầu, có thể lên đến hàng triệu hoặc hàng tỷ đỉnh) thường chiếm phần lớn thời gian chạy của thuật toán. Vì vậy, trong [2], các tác giả chỉ so sánh thời gian chạy của vòng lặp đầu tiên, cụ thể trong hình 2.5, trong đó trục tung là thời gian chạy vòng lặp đầu tiên tính bằng giây, trục hoành là các bộ dữ liệu được sử dụng để kiểm tra thời gian chạy của thuật toán, gồm DBLP, Amazon, IMDB, Live Journal, Web of Science, và Web UK (2005). Cột màu đỏ biểu diễn thời gian chạy của thuật toán Louvain. Cột màu xanh biểu diễn thời gian chạy của thuật toán Leiden.



Hình 2.5: Thời gian chạy vòng lặp đầu tiên của hai thuật toán Louvain và Leiden.

Nhận xét 2.3.3. *Từ biểu đồ trên, chúng ta có các nhận xét sau:*

- *Với tất cả các bộ dữ liệu, thuật toán Louvain (màu đỏ) thường có thời gian chạy lâu hơn so với Leiden (màu xanh).*
- *Sự khác biệt giữa hai thuật toán rõ rệt nhất ở các bộ dữ liệu lớn hơn như Web of Science và Web UK (2005), thời gian của Louvain gấp nhiều lần Leiden.*

2.3.2. Thí nghiệm trên đồ thị ngẫu nhiên

Bên cạnh việc đánh giá thông qua các chỉ số modularity Q_u, chúng tôi còn sử dụng chỉ số Jaccard để đo lường chất lượng phân cụm trên các đồ thị ngẫu nhiên có cấu trúc cộng đồng. Nếu phân cụm từ thuật toán trùng khớp hoàn toàn với phân cụm gốc của đồ thị thì chỉ số Jaccard bằng 1.

Chỉ số Jaccard đánh giá chất lượng cộng đồng mạng

Chỉ số Jaccard [7] đo lường mức độ chồng chéo giữa hai tập hợp. Chỉ số này được tính bằng cách lấy tỷ lệ giữa kích thước giao của hai tập hợp và kích thước hợp của chúng.

Với hai tập hợp A và B, chỉ số Jaccard được tính theo công thức:

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{2.3.1}$$

Giả sử chúng ta có hai phân hoạch của đồ thị G, $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, ...\}$ và $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, ...\}$, chỉ số Jaccard cho từng cặp cụm $\mathcal{H}_i$ và $\mathcal{K}_j$ sẽ được tính. Để tổng quát, chúng ta có thể tính giá trị Jaccard trung bình (MJC) như sau:

$$MJC = \frac{1}{h} \sum_{i=1}^{h} \max_j \text{Jaccard}(\mathcal{H}_i, \mathcal{K}_j). \tag{2.3.2}$$

Đồ thị ngẫu nhiên theo hai mô hình ngẫu nhiên

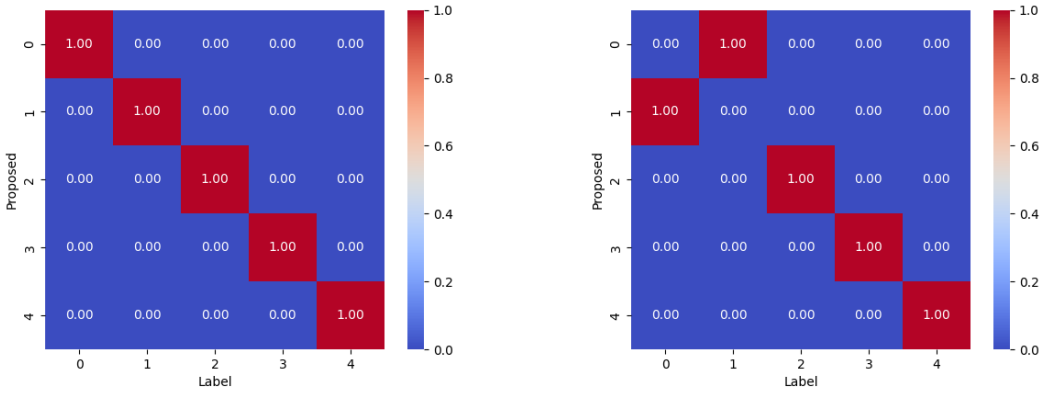
Chúng tôi tạo hai đồ thị ngẫu nhiên theo mô hình phân vùng $l-$ vùng và mô hình phân vùng Gaussian ngẫu nhiên với các thông số như sau:

- Đồ thị G' tuân theo mô hình phân vùng $l-$ vùng với số cụm $g = 5$, số đỉnh mỗi cụm $l = 25$, xác suất xuất hiện cạnh giữa hai đỉnh trong cụm $p_{in} = 0.5$, xác suất xuất hiện cạnh giữa hai đỉnh khác cụm $p_{out} = 0.01$.
- Đồ thị G^* tuân theo mô hình phân vùng Gaussian ngẫu nhiên với tổng số đỉnh $n = 200$, phương sai số đỉnh trong một cụm $\mu = 40$, xác suất xuất hiện cạnh giữa hai đỉnh trong cụm $p_{in} = 0.4$, xác suất xuất hiện cạnh giữa hai đỉnh khác cụm $p_{out} = 0.1$.

Kết quả của việc thực hiện thuật toán Louvain và Leiden trên hai đồ thị G', G^* được thể hiện thông qua bảng 2.5, hình 2.6, và hình 2.7.

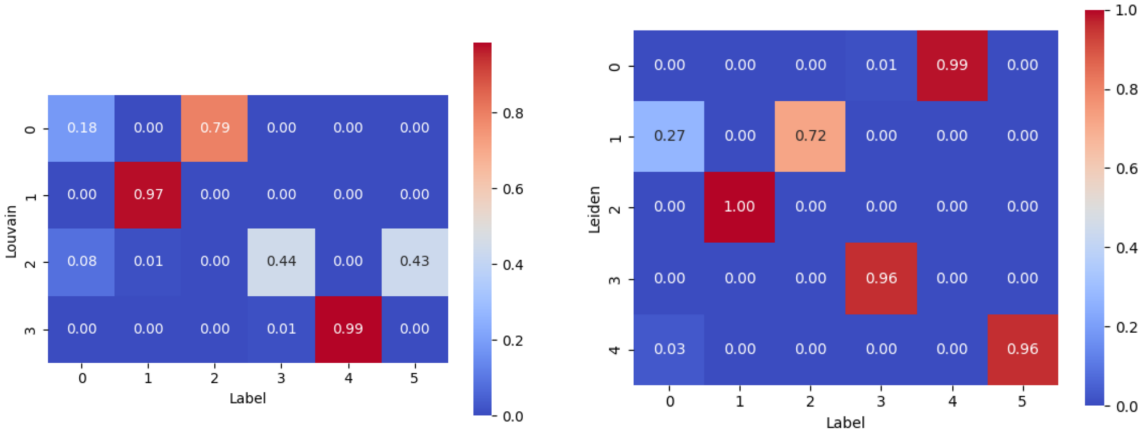
Đồ thị	Thuật toán	Q_u **từ thuật toán**	Q_u **theo nhãn**	**MJC**
G'	Louvain	0.728	0.728	1.000
G'	Leiden	0.728	0.728	1.000
G^*	Louvain	0.263	0.259	0.799
G^*	Leiden	0.264	0.259	0.911

Bảng 2.5: Kết quả thử nghiệm trên đồ thị ngẫu nhiên của hai thuật toán Louvain và Leiden.



(a) Thuật toán Louvain (b) Thuật toán Leiden

Hình 2.6: Kết quả trên đồ thị G': Ma trận nhiệt độ thể hiện chỉ số Jaccard của từng cặp cộng đồng.



(a) Thuật toán Louvain (b) Thuật toán Leiden

Hình 2.7: Kết quả trên đồ thị G^*: Ma trận nhiệt độ thể hiện chỉ số Jaccard của từng cặp cộng đồng.

Nhận xét 2.3.4.

- *Từ thông số xác suất cạnh trong và ngoài cụm của hai đồ thị, ta nhận thấy đồ thị G' có sự liên kết các đỉnh trong cùng một đồ thị là dày đặc (khi $p_{in} = 0.5, p_{out} = 0.01$ trong khi đó đồ thị G^* thì có sự liên kết ít dày đặc hơn khi $p_{in} = 0.4, p_{out} = 0.1$.*
- *Bảng 2.5 minh hoạ kết quả thử nghiệm của hai thuật toán Louvain và Leiden trên hai đồ thị ngẫu nhiên G^*, G'.*
- *Giá trị moudlarity Q_u và giá trị chỉ số Jaccard trung bình (MJC) là hai đại lượng để so sánh với nhãn cộng đồng của các đồ thị ngẫu nhiên.*
- *Dựa trên giá trị Q_u: có thể thấy cả hai thuật toán hoạt động tốt khi giá trị của cách phân chia cộng đồng từ cả hai thuật toán đều lớn hơn hoặc bằng modularity của nhãn cộng đồng.*
- *Mỗi phần tử trên ma trận nhiệt độ của hình 2.6 và 2.7 thể hiện chỉ số Jaccard trên từng cặp cộng đồng giữa cộng đồng phát hiện từ thuật toán và một cộng đồng nhãn dựa trên mô hình.*
- *Dựa trên giá trị MJC và hình 2.7 có thể thấy thuật toán Leiden có kết quả phân cụm gần với nhãn cộng đồng của đồ thị G^* hơn.*

Algorithm 2 Thuật toán Leiden

Input: Đồ thị $G = (V, E)$
Output: Cách chia các cộng đồng $C_1, C_2, \ldots, C_k$

Biến chỉ cấp đang xét: $l = 0$ và khởi tạo $\mathcal{P} = \emptyset$
Đồ thị cấp 0: $G^0 = (V^0, E^0) = (V, E) = G$
while ($l = 0$ *hoặc* $|\mathcal{P}| = |V^l|$) **do**
    /* Xét đồ thị tại cấp l */
    Khởi tạo mỗi đỉnh là một cộng đồng riêng lẻ: $\mathcal{P} = \{\{1\}, \{2\}, ..., \{|V^l|\}\}$
    Khởi tạo hàng đợi $\mathcal{K} = \{1, 2, 3, ..., |V^l|\}$
    /* Giai đoạn 1: Tối ưu modularity */
    while $\mathcal{K} \neq \emptyset$ **do**
        for $v \in \mathcal{K}$ **do**
            /* Loại bỏ v khỏi hàng đợi $\mathcal{K}$ */
            $\mathcal{K}$.remove(v)
            /* Tính độ cải thiện modularity khi $i \to C$ */
            Tập hợp các cộng đồng lân cận: $H = \{C | C \in \mathcal{P}, (i, j) \in E, j \in C\}$
            for $C \in H$ **do**
                Tính $\Delta Q_f^{i \to C}$ bằng công thức (2.2.5)
            end
            if $\max \Delta Q_f^{i \to C} > 0$ **then**
                Đặt $C_{max} = \text{argmax}_C \Delta Q_f^{i \to C}$
                Cập nhật $\mathcal{P}$: Di chuyển đỉnh i vào cộng đồng C_{max}
                Thêm tập $\mathcal{N}$ (như trong công thức (2.2.6)) vào hàng đợi $\mathcal{K}$
            end
        end
    end
    /* Giai đoạn 2: Giai đoạn tinh chỉnh */
    Khởi tạo $\mathcal{P}_{refine} = \{\{1\}, \{2\}, ..., \{|V^l|\}\}$
    for $C \in \mathcal{P}$ **do**
        Xác định tập R như công thức (2.2.8)
        for $v \in R$ **do**
            if $\exists \{v\} \in \mathcal{P}_{refine}$ **then**
                Xác định tập T như công thức (2.2.9)
                Di chuyển đỉnh v theo xác suất như công thức (2.2.10) và cập nhật $\mathcal{P}_{refine}$
            end
        end
    end
    /* Giai đoạn 3: Tổng hợp cộng đồng, tạo đồ thị mới */
    Đặt $\mathcal{V} = \{(C, D) | C, D \in \mathcal{P}_{refine}\}$
    for $(C_x, C_y) \in \mathcal{V}$ **do**
        Tính trọng số cạnh giữa các siêu đỉnh $A_{C_x C_y}$ trong đồ thị mới G^{l+1} theo công thức (2.1.5)
    end
    Gán các siêu đỉnh vào các cụm dựa trên cách phân cụm $\mathcal{P}$ từ giai đoạn 1.
    $l = l + 1$
end
return Cách phân cụm $\mathcal{P}$ với các đỉnh của đồ thị ban đầu (G)

Chương 3

Các thuật toán tìm kiếm cộng đồng mạng sử dụng phương pháp tối ưu modularity toàn cục

Trong chương này, chúng tôi sẽ giới thiệu phương pháp phân tích phổ cho đồ thị vô hướng, dựa trên nghiên cứu của Newman trong bài báo [3]. Đồng thời, dựa trên ý tưởng sử dụng bước đi ngẫu nhiên trên đồ thị, chúng tôi cũng đề xuất một hàm modularity mới và phương pháp phân tích phổ tương ứng cho đồ thị có hướng, kết quả này được viết trong bài báo [32].

3.1. Phương pháp phổ cho đồ thị vô hướng

3.1.1. Phương pháp phổ cho trường hợp hai cộng đồng

Phương pháp phổ của Newman [3] chia đồ thị thành hai cộng đồng dựa trên vector riêng của ma trận Laplacian. Các bước chính trong quá trình thực hiện thuật toán bao gồm:

- **Xác định mục tiêu**: Đầu tiên, chúng ta cần xác định mục tiêu của bài toán, đó là tối ưu hóa việc phân chia các đỉnh của đồ thị thành hai cộng đồng sao cho modularity được tối đa hóa.
- **Chuyển đổi bài toán**: Sau khi xác định được mục tiêu, thuật toán sẽ đưa ra các ràng buộc cho các biến, giúp chuyển bài toán ban đầu (thường phức tạp) thành một bài toán xấp xỉ. Điều này cho phép chúng ta dễ dàng tìm được nghiệm gần đúng.

- **Sử dụng phương pháp Lagrange**: Tiếp theo, phương pháp nhân tử Lagrange được áp dụng để giải bài toán xấp xỉ này, nhằm tìm ra nghiệm tối ưu. Đây là bước quan trọng giúp ta xác định các yếu tố chính ảnh hưởng đến cách phân chia cộng đồng.

- **Tìm vector riêng**: Cuối cùng, nghiệm của bài toán được tìm bằng cách xác định vector riêng ứng với giá trị riêng lớn thứ hai của ma trận liên kết. Dựa vào vector riêng này, các đỉnh của đồ thị sẽ được phân chia thành hai cộng đồng khác nhau một cách hiệu quả.

Nhờ vào việc sử dụng ma trận Laplacian và các kỹ thuật tối ưu hóa, phương pháp phổ của Newman giúp xác định cấu trúc cộng đồng trong mạng một cách nhanh chóng và chính xác.

Tiếp theo, chúng ta sẽ trình bày chi tiết về phương pháp phổ. Cụ thể, mục tiêu của phương pháp là phân chia đồ thị thành hai cộng đồng sao cho giá trị của hàm modularity Q_u được tối ưu hóa (đạt giá trị cực đại). Hàm modularity được định nghĩa như sau:

$$Q_u = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta(g_i, g_j), \tag{3.1.1}$$

Mục tiêu là phân chia đồ thị thành hai cộng đồng C_1 và C_2 sao cho giá trị của Q_u đạt cực đại. Điều này tương đương với việc tìm cách sắp xếp các đỉnh sao cho số cạnh thực tế giữa các đỉnh trong cùng một cộng đồng là nhiều nhất, so với số cạnh kỳ vọng nếu các cạnh được phân phối ngẫu nhiên.

Bài toán tối ưu: Tìm cách phân chia đồ thị thành hai cộng đồng C_1 và C_2 sao cho:

$$\max_{C_1, C_2} Q_u, \tag{3.1.2}$$

Để đơn giản hóa việc giải bài toán tối ưu, chúng ta thay thế biến $\delta(g_i, g_j)$ bằng biến mới s_i như sau:

$$s_i = \begin{cases} 1 & \text{nếu đỉnh } i \text{ thuộc cộng đồng } C_1, \\ -1 & \text{nếu đỉnh } i \text{ thuộc cộng đồng } C_2. \end{cases}$$

Từ đó chúng ta có:

$$\delta(g_i, g_j) = \frac{1}{2}(s_i s_j + 1).$$

Thay vào (3.1.1), ta thu được:

$$Q_u = \frac{1}{4m} \sum_{ij} \left[A_{ij} - \frac{d_i d_j}{2m} \right] (s_i s_j + 1). \tag{3.1.3}$$

Ta đặt B là ma trận modularity được xác định như sau:

$$B_{ij} = A_{ij} - \frac{d_i d_j}{2m} \tag{3.1.4}$$

Suy ra:

$$Q_u = \frac{1}{4m} \sum_{ij} B_{ij}(s_i s_j + 1), \tag{3.1.5}$$

Dễ thấy:

$$\sum_j B_{ij} = \sum_j A_{ij} - \frac{d_i}{2m} \sum_j d_j = d_i - \frac{d_i}{2m} \times 2m = 0, \tag{3.1.6}$$

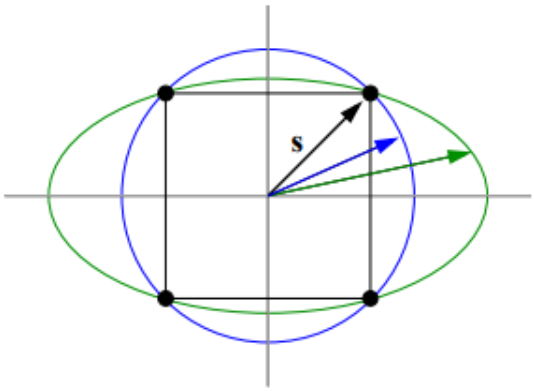
Thay (3.1.6) vào (3.1.5), ta được:

$$Q_u = \frac{1}{4m} \sum_{ij} B_{ij} s_i s_j. \tag{3.1.7}$$

Từ đó bài toán gốc (3.1.2) được viết lại với các biến $s = (s_1, s_2, ..., s_N)^T$ như sau:

$$\max_s Q_u = \max_s \frac{1}{4m} \sum_{ij} B_{ij} s_i s_j \quad \text{với điều kiện: } s_i \in \{\pm 1\} \quad \forall i = 1, 2, \ldots, n, \tag{3.1.8}$$

Để giải chính xác bài toán tối ưu với các biến s_i thuộc tập giá trị rời rạc, bài toán này được xếp vào loại NP-khó (theo nghiên cứu [33]). Vì vậy, Newman đã chuyển đổi nó thành một bài toán xấp xỉ. Thay vì chỉ tìm nghiệm trong tập các giá trị rời rạc, phương pháp sẽ tìm nghiệm xấp xỉ trên các miền là các siêu cầu hoặc siêu elip, mà chúng đi qua các điểm rời rạc đó. Xem minh hoạ ở Hình 3.1 (nguồn: bài báo [3]).



Hình 3.1: Hình minh họa quá trình chuyển đổi từ tập giá trị rời rạc sang miền liên tục.

Từ đó, thay vì chỉ tìm nghiệm trên tập rời rạc $s_i \in \{\pm 1\}$, Newman đề xuất việc xấp xỉ bằng cách thay thế tập giá trị rời rạc bằng một đường cong elip với điều kiện như sau:

$$\sum_i d_i s_i^2 = \sum_i d_i = 2m.$$

Trong đó, d_i là bậc của đỉnh i.

Như vậy, thay vì nhận được kết quả chính xác $s_i = 1$ (tương ứng với đỉnh i thuộc cộng đồng C_1) hoặc $s_i = -1$ (tương ứng với đỉnh i thuộc cộng đồng C_2), bài toán xấp xỉ sẽ cung cấp giá trị s_i là một số thực. Để xác định cộng đồng của đỉnh i, chúng ta sẽ áp dụng quy tắc: nếu s_i dương, ta gán $s_i = 1$ (đỉnh thuộc cộng đồng C_1), và nếu s_i âm, ta gán $s_i = -1$ (đỉnh thuộc cộng đồng C_2).

Từ đó, chúng ta thu được bài toán tối ưu trên miền liên tục như sau:

$$\max_s \frac{1}{4m} \sum_{ij} B_{ij} s_i s_j \quad \text{với điều kiện: } \sum_i^n d_i s_i^2 = 2m, \tag{3.1.9}$$

Để áp dụng Định lý 1.5.5 tìm giá trị nhỏ nhất của bài toán tối ưu có ràng buộc, hàm mục tiêu được biểu diễn dưới dạng:

$$f(s) = \frac{-1}{4m} \sum_{ij} B_{ij} s_i s_j,$$

và điều kiện ràng buộc là:

$$h(s) = \sum_i d_i s_i^2 - 2m = 0.$$

Phương pháp nhân tử Lagrange biến bài toán tối ưu trở thành:

$$\mathcal{L}(s,\mu) = f(s) + \mu h(s) = \frac{-1}{4m}\sum_{ij} B_{ij}s_i s_j + \mu\left(\sum_i d_i s_i^2 - 2m\right).$$

Bằng cách tạo các đại lượng $f(s), h(s)$ như trên, ta đã quy bài toán (3.1.9) thành bài toán tối ưu như trong Định lý 1.5.5, sử dụng điều kiện thứ hai của định lý này, nghiệm tối ưu của bài toán (3.1.9) cần thoả mãn phương trình sau:

$$\nabla_s \mathcal{L} = \begin{bmatrix} \frac{\partial}{\partial s_1}\left(\frac{-1}{4m}\sum_{ij} B_{ij}s_i s_j + \mu\left(\sum_i d_i s_i^2 - 2m\right)\right) \\ \vdots \\ \frac{\partial}{\partial s_N}\left(\frac{-1}{4m}\sum_{ij} B_{ij}s_i s_j + \mu\left(\sum_i d_i s_i^2 - 2m\right)\right) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\Leftrightarrow \frac{-1}{4m}\sum_j B_{ij}s_j + 2\mu d_i s_i = 0.$$

Đặt $\lambda = 2\mu$, ta thu được:

$$\sum_j B_{ij}s_j = \lambda d_i s_i \tag{3.1.10}$$

Viết phương trình (3.1.10) dưới dạng ma trận, ta có phương trình sau:

$$Bs = \lambda Ds, \tag{3.1.11}$$

Do đó, giá trị cực đại của hàm modularity Q là:

$$Q_{max} = \frac{1}{4m}\sum_{ij} B_{ij}s_i s_j = \frac{\lambda}{4m}\sum d_i s_i^2 = \frac{\lambda}{2}. \tag{3.1.12}$$

Nhân cả hai vế của phương trình $Bs = \lambda Ds$ với vector $\mathbf{1}^T = (1,1,...,1)^T$, ta có:

$$As = \lambda Ds. \tag{3.1.13}$$

Dễ thấy rằng $s = (1,1,\ldots,1)^T$ là vector riêng ứng với giá trị riêng $\lambda = 1$, điều này có nghĩa là tất cả các đỉnh đều thuộc cùng một cụm, không phân biệt được cấu trúc cộng đồng trong đồ thị. Theo định lý Perron–Frobenius [3], giá trị riêng lớn nhất của một ma trận không âm sẽ có một vector riêng tương ứng mà tất cả các thành phần đều dương, và trong trường hợp này, đó là vector $(1,1,\ldots,1)^T$. Tuy nhiên, do vector riêng này không cung cấp thông tin hữu

ích cho việc phát hiện cộng đồng, ta cần loại trừ nó. Thay vào đó, ta chọn vector riêng ứng với giá trị riêng lớn thứ hai.

Chúng ta có thể biểu diễn lại nghiệm thông qua ma trận Laplacian chuẩn hóa:

$$L = D^{-1/2} A D^{-1/2}, \tag{3.1.14}$$

trong đó D là ma trận bậc của đồ thị. Ta có:

$$u = D^{1/2} s, \quad \Rightarrow \left(D^{-1/2} A D^{-1/2} \right) u = \lambda u \quad \Leftrightarrow \quad Lu = \lambda u. \tag{3.1.15}$$

Bởi vì các phần từ trên đường chéo trong $D^{1/2}$ đều lớn hơn 0 nên phần tử thứ i của vector u sẽ cùng dấu với phần tử thứ i trong vector s, do vậy, ta có thể xác định nghiệm thông qua vector riêng u. Cụ thể, dựa trên dấu của các phần tử trên vector riêng u, ta phân thành tương ứng hai cộng đồng.

Từ phân tích ở trên, ta có được phương pháp phổ chia đồ thị vô hướng thành 2 cộng đồng gồm 2 bước:

- Bước 1: Tính giá trị vector riêng u ứng với giá trị riêng cao thứ 2 của ma trận Laplacian chuẩn hoá trong công thức (3.1.15).
- Bước 2: Chia các đỉnh của đồ thị thành 2 cộng đồng ứng với dấu của các phần tử

3.1.2. Phương pháp phổ cho đồ thị vô hướng cho trường hợp nhiều hơn hai cộng đồng

Nghiệm của phương trình (3.1.13) tương ứng với kết quả chia đồ thị thành 2 cộng dồng dựa trên dấu của các phần tử trong vector riêng của ma trận Laplacian chuẩn hoá. Trong phần này, chúng tôi áp dụng một cơ chế tự động chia đồ thị thành nhiều hơn hai cộng đồng. Ý tưởng chính của thuật toán là áp dụng phương pháp phổ lên đồ thị gốc, sau đó thu được hai cộng đồng C_1, C_2. Từ đây ta tạo hai đồ thị cảm sinh G_1, G_2 từ hai cộng đồng này và thực hiện phương pháp phổ trên từng đồ thị cảm sinh. Quá trình này được thực hiện cho tới khi việc phân chia thành hai phần nhỏ hơn không thể cải thiện modularity. Chi tiết được mô tả trong phần mã giả của Thuật toán 3.

Algorithm 3 Thuật toán chia đồ thị vô hướng thành các cộng đồng

Input: Đồ thị $G = (V, E)$, ngưỡng cải thiện modularity ϵ
Output: Phân hoạch các cộng đồng $C_1, C_2, \ldots, C_k$

Khởi tạo tập các cộng đồng $\mathcal{C} = \{G\}$
while $\exists C_i \in C$ *chưa được xét* **do**
    `/* Xét cộng đồng` C_i `chưa được xét:` */
    `/* Bước 1: Chia thành 2 cụm bằng phương pháp phổ` */
    Sử dụng phương pháp phổ, xác định hai cộng đồng con C_{1i}, C_{2i} từ C_i;
    `/* Bước 2: Kiểm tra giá trị modularity` */
    Tính độ cải thiện modularity ΔQ_u với trường hợp chia $\{C_i\} \to \{C_{1i}, C_{2i}\}$
    if $\Delta Q_u \geq \epsilon$ **then**
        Cập nhật phân hoạch: $\mathcal{C} = \mathcal{C} \setminus \{C_i\} \cup \{C_{1i}, C_{2i}\}$
    end
    else
        Giữ nguyên cộng đồng C_i và đánh dấu là đã được xét
    end
end
return phân hoạch cuối cùng $\mathcal{C}$

3.1.3. Độ phức tạp tính toán

Từ mô tả thuật toán và mã giả tương của Thuật toán 3 tương ứng, dễ thấy đây là thuật toán chia để trị. Cụ thể, ta có tính độ phức tạp của cả thuật toán (kí hiệu là $T(n)$) sau khi làm rõ từng bước dưới đây:

- Chia: là bước để chia dữ liệu có kích thước n thành dữ liệu nhỏ hơn - tương ứng với "Bước 1: Chia thành 2 cụm bằng phương pháp phổ", bước này chủ yếu dựa trên việc tính vector riêng, giá trị riêng nên độ phức tạp tương ứng với phép nhân ma trận, ở đây chọn cách nhân thông thường là $O(n^3)$.

- Trị: là bước thời gian chạy tổng hợp với các phần dữ liệu nhỏ hơn, tổng quát thì phần này có độ phức tạp là $T(|C_1|) + T(|C_2|)$. Đặt $k = |C_1|$ thì ta biểu diễn lại độ phức tạp tại bước này là: $T(k) + T(n-k)$.

- Tổng hợp: là bước tổng hợp kết quả, trong bài toán này thì bước này không mất thời gian.

Vậy ta có phương trình đệ qui về độ phức tạp tính toán của Thuật toán 3:

$$T(n) = \begin{cases} \Theta(1) & \text{nếu } n = 1 \\ T(k) + T(n-k) + O(n^3) & \text{nếu } n > 1 \end{cases} \tag{3.1.16}$$

Tại trường hợp hai cộng đồng được phát hiện có kích thước như nhau thì phương trình (3.1.16) trở thành:

$$T(n) = 2T(n/2) + O(n^3), \tag{3.1.17}$$

Còn trong trường hợp xấu nhất, qua mỗi lần chia chỉ phát hiện được một cộng đồng có kích thước là một đỉnh thì phương trình (3.1.16) trở thành:

$$T(n) = T(n-1) + O(n^3), \tag{3.1.18}$$

Sử dụng định lý Master [34] để tính độ phức tạp của $T(n)$ trong trường hợp (3.1.17) là $O(n^3)$; còn trong trường hợp (3.1.18) thì ta có thể tính đơn giản:

$$T(n) = T(n-1) + O(n^3) = T(n-2) + O(n^3) + O((n-1)^3)$$

$$= T(1) + O(\sum_{i=2}^{n}(i^3)) = O(n^4).$$

3.2. Phương pháp phổ cho đồ thị có hướng

3.2.1. Định nghĩa hàm modularity cho đồ thị có hướng

Trong phần này, chúng tôi đề xuất một định nghĩa mới cho hàm modularity Q_p trên đồ thị có hướng, dựa trên các bước đi ngẫu nhiên và phân phối dừng. Đặc biệt, định nghĩa này vẫn có thể áp dụng cho đồ thị vô hướng, mang lại tính linh hoạt cao trong việc sử dụng hàm modularity cho các loại đồ thị khác nhau.

Trước tiên, chúng ta nhắc lại hàm modularity cho đồ thị vô hướng được đề xuất bởi Newman [9]:

$$Q_u = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{d_i d_j}{2m}\right]\delta(g_i, g_j), \tag{3.2.1}$$

Trong đó, A_{ij} là ma trận kề biểu thị số cạnh giữa đỉnh i và j, d_i và d_j là bậc của các đỉnh i và j, và $2m$ là tổng số cạnh của đồ thị.

Tiếp theo, khi sử dụng phân phối dừng của quá trình bước đi ngẫu nhiên trên đồ thị vô hướng, ta có ma trận xác suất chuyển tiếp $P = D^{-1}A$, với D là ma trận đường chéo chứa bậc của các đỉnh. Khi đó, A_{ij} có thể viết lại dưới dạng $P_{ji}d_j$. Nhờ vậy, hàm Q_u có thể được biểu diễn lại như sau:

$$Q_u = \sum_{ij} \left[P_{ji}\frac{d_j}{2m} - \frac{d_i}{2m}\frac{d_j}{2m} \right] \delta(g_i, g_j). \tag{3.2.2}$$

Giả sử điều kiện tồn tại phân phối dừng 1.2.2 được thỏa mãn, phân phối dừng ϕ_i sẽ có dạng $\phi_i = \frac{d_i}{2m}$ (xem trong [35]), trong đó d_i là bậc của đỉnh i và $2m$ là tổng số cạnh. Khi đó, công thức modularity có thể biến đổi thành:

$$Q_u = \sum_{ij} \left[P_{ji}\phi_j - \phi_i\phi_j \right] \delta(g_i, g_j), \tag{3.2.3}$$

Từ cách tiếp cận này, chúng tôi định nghĩa hàm modularity Q_p cho đồ thị có hướng, tương tự như trường hợp đồ thị vô hướng, nhưng dựa trên các xác suất chuyển tiếp giữa các đỉnh. Định nghĩa cụ thể như sau:

$$Q_p = \sum_{ij} \left[P_{ji}\phi_j - \phi_i\phi_j \right] \delta(g_i, g_j), \tag{3.2.4}$$

Trong đó:

- P_{ji} là xác suất chuyển từ đỉnh j sang đỉnh i,
- ϕ_i là thành phần thứ i trong phân phối dừng ϕ, đại diện cho xác suất mà bước đi ngẫu nhiên dừng lại ở đỉnh i.

Định nghĩa này hàm ý rằng các đỉnh trong cùng một cụm nên có xác suất di chuyển qua lại lớn hơn so với xác suất dừng ngẫu nhiên tại các đỉnh riêng lẻ. Cụ thể, nếu hai đỉnh i và j cùng thuộc một cụm, ta kỳ vọng rằng xác suất chuyển P_{ji} giữa chúng lớn hơn so với xác suất dừng ϕ_i, tức là $P_{ji}\phi_j > \phi_i\phi_j$. Điều này phản ánh tính kết nối chặt chẽ giữa các đỉnh trong cùng cụm thông qua các bước đi ngẫu nhiên.

Ý nghĩa từ góc nhìn xác suất: Thật vậy, với kí hiệu $\{X_k\}_{k=1,2,...}$ là quá trình bước đi ngẫu nhiên trên đồ thị với X_k là vị trí của bước đi sau k bước, ta có:

$$P_{ji} = P(X_{k+1} = i | X_k = j), \tag{3.2.5}$$

thì ta có xác suất di chuyển từ đỉnh $j \to i$ là P_{ji} trong khi đó theo công thức (3.2.4) thì ϕ_i chính là xác suất để quá trình bước đi ngẫu nhiên "dừng" tại đỉnh i bất kể bắt đầu từ đâu. Do đó, chúng tôi nhận thấy nếu đỉnh j, i cùng cụm thì giá trị $P_{ji} > \phi_i$ hay $\phi_j P_{ji} > \phi_i \phi_j$. Kết quả này tương tự với biểu thức $\phi_i P_{ij} > \phi_i \phi_j$. Hơn thế nữa, các cặp biểu thức trong hàm Q_p cũng có ý nghĩa về mặt xác suất:

$$P_{ji}\phi_j = P_{ji}.\lim_{k\to\infty} P\left(X_k = j\right),$$
$$\Leftrightarrow P_{ji}\phi_j = \lim_{k\to\infty} P\left(X_k = j\right) P(X_{k+1} = i | X_k = j),$$

Vậy ta thu được biểu thức $P_{ji}\phi_j$ chính là xác suất đồng thời để cặp trạng thái hay cặp đỉnh (i, j) tại thời điểm "dừng".

$$P_{ji}\phi_j = \lim_{k\to\infty} P\left(X_{k+1} = i, X_k = j\right), \tag{3.2.6}$$

Tương tự, ta cũng thu được:

$$P_{ij}\phi_i = \lim_{k\to\infty} P\left(X_{k+1} = j, X_k = i\right), \tag{3.2.7}$$

Trong khi đó thì theo công thức (3.2.4) thì biểu thức:

$$\phi_i\phi_j = \lim_{k\to\infty} P\left(X_{k+1} = i\right) \times \lim_{k\to\infty} P\left(X_k = j\right) = \lim_{k\to\infty} P\left(X_k = i\right) \times \lim_{k\to\infty} P\left(X_{k+1} = j\right)$$

Về mặt ý nghĩa, tính hợp lý của việc đề xuất hai biểu thức có thể giải thích như sau: bởi chúng tôi quan tâm tới thời điểm đặc biệt của quá trình bước đi ngẫu nhiên được áp dụng trên đồ thị ban đầu, ở đây là thời điểm "dừng", do vậy nếu hai đỉnh(trạng thái) i, j có quan hệ chặt chẽ thì sự kiện "bước đi ngẫu nhiên đang ở đỉnh j" $(X_k = j)$ sẽ có xu hướng đi tới "đỉnh i" $(X_{k+1} = i)$ (và ngược lại) có xác suất xảy ra lớn hơn việc hai sự kiện đó xảy ra độc lập.

Về mặt toán học, ta đặt hai sự kiện như sau:

- Sự kiện I_1:"tại thời điểm quá trình đã dừng, bước đi ngẫu nhiên tại thời điểm k đang ở đỉnh j"

- Sự kiện I_2:"tại thời điểm quá trình đã dừng, bước đi ngẫu nhiên tại thời điểm $k+1$ đang ở đỉnh j"

Theo phương trình 3.2.6 thì $P_{ji}\phi_j = P(I_1, I_2)$ trong khi đó $\phi_i\phi_j = P(I_1)P(I_2)$. Với hai sự kiện là có liên quan thì khi $P(I_1, I_2) > P(I_1)P(I_2)$, điều này cho thấy hai sự kiện có xu hướng xảy ra cùng nhau nhiều hơn so với trường hợp chúng độc lập. Đây có thể là dấu hiệu của một mối liên quan tích cực giữa hai sự kiện này.

3.2.2. Phương pháp phổ cho đồ thị có hướng

Phân chia thành hai cộng đồng

Hàm modularity Q_p mà chúng tôi đề xuất [32] không chỉ hợp lý về mặt ý nghĩa và thống nhất với hàm Q_u mà còn có thể được tối ưu toàn cục bằng phương pháp phổ tương tự như của Newman. Phương pháp này dựa trên việc phân chia đồ thị thành hai cộng đồng C_1 và C_2.

Bài toán tối ưu: Tìm cách phân chia đồ thị thành hai cộng đồng C_1 và C_2 sao cho:

$$\max_{C_1,C_2} Q_p, \tag{3.2.8}$$

Trước tiên, chúng tôi thay thế biểu thức $\delta(g_i, g_j)$ bằng biến đơn giản hơn:

$$s_i = \begin{cases} 1 & \text{nếu } i \in C_1, \\ -1 & \text{nếu } i \in C_2. \end{cases}$$

Khi đó, hàm Q_p có thể được biểu diễn lại như sau:

$$Q_p = \frac{1}{2}\sum_{ij}\left(P_{ji}\phi_j - \phi_i\phi_j\right)(s_i s_j + 1). \tag{3.2.9}$$

Để đơn giản tính toán, chúng tôi định nghĩa ma trận M như sau:

$$M_{ij} = P_{ji}\phi_j - \phi_i\phi_j. \tag{3.2.10}$$

Từ đó, hàm Q_p trong công thức (3.2.9) có thể được biểu diễn ngắn gọn như sau:

$$Q_p = \frac{1}{2}\sum_{ij} M_{ij}(s_i s_j + 1). \tag{3.2.11}$$

Vì tổng của các phần tử trong ma trận M là 0, ta có:

$$\sum_{i,j} M_{ij} = 0,$$

do đó hàm Q_p trong phương trình (3.2.11) trở thành:

$$Q_p = \frac{1}{2} \sum_{ij} M_{ij} s_i s_j. \tag{3.2.12}$$

Bài toán tối ưu có thể được diễn đạt lại dưới dạng:

$$\max_s Q_p = \max_s \frac{1}{2} \sum_{ij} M_{ij} s_i s_j, \tag{3.2.13}$$

với $s_i \in \{-1, 1\}$, tức là mỗi đỉnh chỉ có thể thuộc một trong hai cộng đồng.

Tương tự như bài toán tối ưu hàm Q_u của Newman, bài toán tối ưu hàm Q_p cũng là một bài toán NP-khó do tính chất rời rạc của tập giá trị s_i. Để giải quyết vấn đề này, chúng tôi sẽ xấp xỉ bài toán bằng cách cho phép các giá trị s_i nằm trong một miền liên tục và áp dụng phương pháp nhân tử Lagrange.

Chúng tôi áp dụng ràng buộc:

$$\sum_i \phi_i s_i^2 = 1,$$

với ϕ_i là thành phần của phân phối dừng. Bài toán xấp xỉ được biểu diễn lại như sau:

$$\max_s \sum_{ij} M_{ij} s_i s_j, \tag{3.2.14}$$

với ràng buộc $\sum_i \phi_i s_i^2 = 1$. Chúng tôi sử dụng nhân tử Lagrange μ để giải bài toán này:

$$\mathcal{L}(s, \mu) = -\sum_{ij} M_{ij} s_i s_j + \mu \left(\sum_i \phi_i s_i^2 - 1 \right). \tag{3.2.15}$$

Lấy đạo hàm riêng của $\mathcal{L}$ theo s_i, ta có:

$$-\sum_j M_{ij} s_j + 2\mu \phi_i s_i = 0. \tag{3.2.16}$$

Đặt $\lambda = 2\mu$, phương trình trên có thể viết lại thành:

$$\sum_j M_{ij} s_j = \lambda \phi_i s_i. \tag{3.2.17}$$

Phương trình (3.2.17) có thể được biểu diễn dưới dạng ma trận:

$$Ms = \lambda \Phi s, \tag{3.2.18}$$

trong đó Φ là ma trận đường chéo với các phần tử ϕ_i.

Giải pháp trên có thể được đơn giản hóa hơn nữa. Bằng cách kết hợp phương trình (3.2.17) và định nghĩa ma trận M:

$$\sum_j P_{ji}\phi_j s_j = \lambda \phi_i s_i + \sum_j \phi_i \phi_j s_j. \tag{3.2.19}$$

Phương trình (3.2.19) có thể được biểu diễn dưới dạng ma trận với $\mathbf{1} = (1, 1, ..., 1)^T \in R^{n\times 1}$:

$$P^T \Phi s = \Phi \left(\lambda s + \left(\boldsymbol{\phi^T s} \right) \mathbf{1} \right). \tag{3.2.20}$$

Chúng ta sẽ đơn giản hóa phương trình (3.2.20) với phép biến đổi sau (chú ý $\mathbf{1} = (1, 1, ..., 1)$):

$$1^T P^T \Phi s = 1^T \Phi \left(\lambda s + \left(\phi^T s \right) \mathbf{1} \right).$$

Vì P là ma trận xác suất chuyển tiếp, do đó

$$1^T P^T = \left(\sum_{i=1}^{N} P_{1i}, \sum_{i=1}^{N} P_{2i}, ..., \sum_{i=1}^{N} P_{ni} \right) = (1, 1, ..., 1) = 1^T,$$

Do đó $1^T P^T \Phi s = 1^T \Phi s = \phi^T$ với $\phi^T \mathbf{1} = \mathbf{1}$, từ đây ta thu được:

$$(\lambda - 1)\, \phi^T s = -\phi^T \left(\phi^T s \right) \mathbf{1},$$

Ta có $\phi^T \left(\phi^T s \right) \mathbf{1} = \phi^T s$, do đó, phương trình trở thành:

$$\lambda \phi^T s = 0.$$

Quay lại công thức (3.2.17), chúng ta có thể thấy rằng nếu $\lambda = 0$ thì thu được:

$$\sum_j M_{ij} s_j = 0, \tag{3.2.21}$$

Phương trình (3.2.21) có một nghiệm là $s = \mathbf{1}$, điều này có nghĩa là tất cả các dấu của phần tử đều dương và tất cả các đỉnh có cùng cộng đồng. Điều này không hữu ích khi có được kết quả đó. Vì vậy, điều kiện của λ là $\lambda \neq 0$. Sau đó, chúng ta có:

$$\phi^T s = 0.$$

Do đó, phương trình (3.2.20) có thể được viết lại đơn giản hơn:

$$P^T \Phi s = \lambda \Phi s. \tag{3.2.22}$$

Nhân cả hai vế của phương trình (3.2.22) với $\Phi^{-1/2}$ và giả sử $u = \Phi^{1/2} s$, chúng ta có:

$$\Phi^{-1/2} P^T \Phi^{1/2} u = \lambda u. \tag{3.2.23}$$

Đặt $L_I = \Phi^{1/2} P \Phi^{-1/2}$, chúng ta có:

$$(L_I)^T u = \lambda u. \tag{3.2.24}$$

Giải phương trình này tương ứng với việc tìm giá trị riêng và vector riêng của ma trận L_I^T. Vector riêng tương ứng với giá trị riêng lớn thứ hai (ngoài 1) sẽ được sử dụng để xác định cộng đồng. Nếu s_i có dấu dương, đỉnh i thuộc cộng đồng C_1, ngược lại, nếu s_i có dấu âm, đỉnh i thuộc cộng đồng C_2.

Từ phân tích trên, ta có được phương pháp phổ chia đồ thị có hướng thành 2 cộng đồng gồm 3 bước:

- Bước 1: Tạo đồ thị G' thoả mãn điều kiện tồn tại phân phối dừng ϕ, trong trường hợp G chưa thoả mãn:
  - Điều kiện liên thông mạnh: thêm cạnh nối giữa các lớp liên thông,
  - Điều kiện chu kỳ bằng 1: thêm 1 cạnh khuyên bất kì.
- Bước 2: Tính giá trị vector riêng u ứng với giá trị riêng thực cao nhất (khác 1) của ma trận Laplacian chuẩn hoá trong công thức (3.2.24).
- Bước 3: Chia các đỉnh thành 2 cộng đồng ứng với dấu của các phần tử.

Tổng quát cho nhiều cộng đồng

Phương pháp phổ này cũng có thể mở rộng để phân chia đồ thị thành nhiều hơn hai cộng đồng. Chi tiết thuật toán như phần mã giả Thuật toán 4.

Algorithm 4 Thuật toán chia đồ thị có hướng thành các cộng đồng.

Input: Đồ thị $G = (V, E)$, ngưỡng cải thiện modularity ϵ
Output: Phân hoạch các cộng đồng $C_1, C_2, \ldots, C_k$

Khởi tạo tập các cộng đồng $\mathcal{C} = \{G\}$
Tập các cộng đồng chưa được xét $\mathcal{Q} = \{\mathcal{C}\}$
while $\exists C_i \in \mathcal{Q}$ **do**
    /* Xét cộng đồng C_i chưa được xét: */
    /* Bước 1: kiểm tra điều kiện tồn tại ϕ */
    if *G không thoả mãn điều kiện tồn tại ϕ (Định nghĩa 1.2.2)* **then**
        /* 1.1: Điều kiện liên thông mạnh */
        Thêm các cạnh giữa các lớp liên thông
        /* 1.2: Điều kiện chu kỳ bằng 1 */
        Thêm cạnh khuyên bất kì
    end
    /* Bước 2: Chia thành 2 cụm bằng phương pháp phổ */
    Sử dụng thuật toán cho 2 cộng đồng, xác định hai cộng đồng con C_{1i}, C_{2i} từ C_i;
    /* Bước 3: Kiểm tra giá trị modularity */
    Tính độ cải thiện modularity ΔQ_p với trường hợp chia $\{C_i\} \to \{C_{1i}, C_{2i}\}$
    if $\Delta Q_p \geq \epsilon$ **then**
        Cập nhật phân hoạch: $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C_i\} \cup \{C_{1i}, C_{2i}\}$
        Cập nhật danh sách cộng đồng cần xét: $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{C_i\} \cup \{C_{1i}, C_{2i}\}$
    end
    else
        Giữ nguyên cộng đồng C_i
        $\mathcal{Q} \leftarrow \mathcal{Q} \setminus C_i$
    end
end
return phân hoạch cuối cùng $\mathcal{C}$

3.2.3. Độ phức tạp tính toán

Cách tính độ phức tạp tính toán trong Thuật toán 4 về cơ bản là giống Thuật toán 3 khi cả hai đều là thuật toán chia để trị. Điểm khác biệt là Thuật toán 4 có xử lý thêm để tồn tại phân phối dừng ϕ.

Khi tính toán phân phối dừng ϕ cho đồ thị có hướng, chúng tôi sử dụng các thuật toán hiệu quả như Thuật toán 1 và Thuật toán 4 trong [36], với độ

phức tạp tính toán là $\tilde{O}(n^{7/4})$ đối với đồ thị thưa.

Đặt độ phức tạp của Thuật toán 4 là $T_1(n)$, độ phức tạp tính toán cụ thể từng bước như sau:

- Chia: bao gồm cả việc tìm phân phối dừng ϕ và tìm giá trị riêng, vector riêng nên tổng độ phức tạp là $O(n^3) + D(n)$ với $D(n)$ là độ phức tạp tính toán phân phối dừng ϕ.
- Trị: vẫn là $T_1(k) + T_1(n-k)$.
- Tổng hợp: không mất thời gian.

Vậy ta có phương trình đệ qui về độ phức tạp tính toán của Thuật toán 4:

$$T_1(n) = \begin{cases} \Theta(1) & \text{nếu } n = 1 \\ T_1(k) + T_1(n-k) + O(n^3) + D(n) & \text{nếu } n > 1 \end{cases} \tag{3.2.25}$$

Trong trường hợp không sử dụng thuật toán nhanh thì mà sử dụng cách tính giải phương trình thông thường để tìm ra Φ thì $D(n) = O(n^3)$ và khi đó thì độ phức tạp giữa hai thuật toán 3 và 4 là như nhau ($T_1(n) = T(n)$). Còn trong trường hợp sử dụng các thuật toán tính nhanh như trong [36] thì độ phức tạp sẽ giảm đi trong trường hợp đồ thị thưa.

3.3. Một số thí nghiệm

Trong phần này, luận văn sẽ trình bày một số thí nghiệm chứng tỏ tính hiệu quả của phương pháp phổ trên đồ thị vô hướng của Newman và phương pháp phổ trên đồ thị có hướng của chúng tôi. Cả hai thuật toán đều được áp dụng trên các đồ thị sinh ngẫu nhiên và đồ thị trong thực tế.

Đối với phương pháp phổ trên đồ thị vô hướng, Newman áp dụng trên đồ thị ngẫu nhiên có cấu trúc hai cộng đồng, ngoài ra thuật toán cũng được áp dụng trên một số đồ thị thực nhỏ.

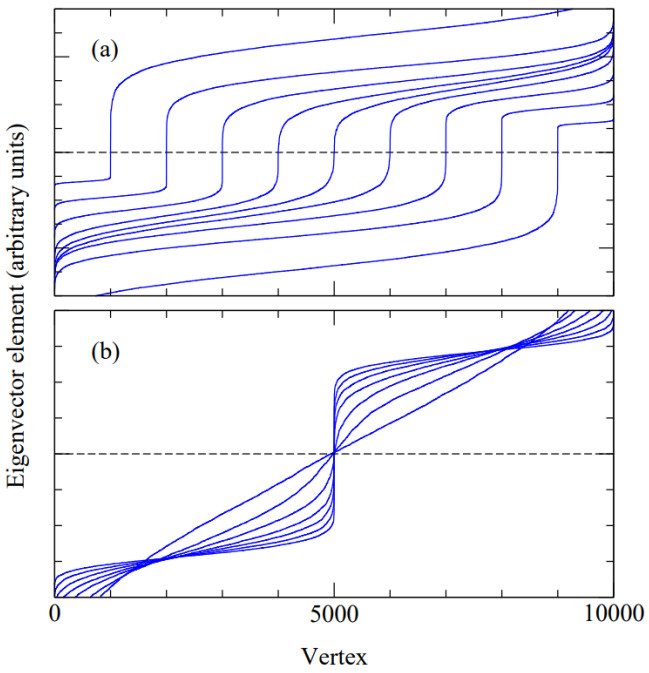
Bên cạnh đó, trong trường hợp có hướng, chúng tôi tạo các đồ thị ngẫu nhiên dựa trên hai mô hình phân vùng ngẫu nhiên được trình bày tại chương 1: mô hình phân vùng $l-$vùng và phân vùng ngẫu nhiên Gaussian. Sau đó,

đánh giá chất lượng phân cụm so với nhãn cộng đồng. Ngoài ra, chúng tôi cũng so sánh phương pháp phổ đề xuất với thuật toán Louvain có hướng [19] trên một số đồ thị thực.

3.3.1. Thí nghiệm trên đồ thị vô hướng

Các thí nghiệm trên đồ thị vô hướng dưới đây được trình bày trong bài báo [3] ứng với thuật toán phương pháp phổ chia đồ thị vô hướng thành hai cộng đồng đã được trình bày tại phần 3.1.1.

Thí nghiệm trên đồ thị ngẫu nhiên



Hình 3.2: Kết quả vector riêng dùng để phân thành hai cộng đồng của các đồ thị ngẫu nhiên.

Trong bài báo [3], các đồ thị ngẫu nhiên được tạo ra từ mô hình khối ngẫu nhiên [37]. Mô hình khối ngẫu nhiên (SBM) được định nghĩa bởi các tham số: số lượng đỉnh n, phân hoạch của tập hợp các đỉnh $\{1,\ldots,n\}$ thành các tập con rời rạc $C_1,\ldots,C_r$ gọi là các cộng đồng, và một ma trận đối xứng kích thước $r \times r$ biểu diễn xác suất có cạnh P giữa các cộng đồng. Các cạnh trong

đồ thị được sinh ngẫu nhiên theo quy tắc: bất kỳ hai đỉnh $u \in C_i$ và $v \in C_j$ sẽ được kết nối với xác suất P_{ij}.

Các kết quả thí nghiệm được trình bày trong hình 3.2 (nguồn: bài báo [3]). Trong đó trục hoành tương ứng thứ tự từ 1 đến 10000 tương ứng với 10000 đỉnh của đồ thị, còn trục tung ứng với giá trị của các phần tử của vector riêng ứng với giá trị riêng thứ 2. Các đường cong màu xanh biểu thị giá trị của các phần tử trong vector riêng này, với đường nét đứt màu đen ngang ứng với giá trị 0. Phần đường cong nằm dưới đường nét đứt đại diện cho các đỉnh thuộc cộng đồng 1, trong khi phần nằm trên đường nét đứt đại diện cho các đỉnh thuộc cộng đồng 2.

Trong thí nghiệm 1: Thí nghiệm này tạo ra 9 đồ thị dựa trên mô hình Stochastic Block Model (SBM) với tổng số 10.000 đỉnh, mỗi đồ thị được chia thành hai cộng đồng. Số lượng đỉnh của cộng đồng 1 dao động từ 1.000 đến 9.000, và xác suất xuất hiện cạnh bên trong mỗi cộng đồng cao gấp 3 lần so với giữa các cộng đồng.

Thuật toán phổ chia đồ thị thành hai cộng đồng và Hình 3.2a ghi lại kết quả vector riêng tương ứng với giá trị riêng thứ hai trong phương trình (3.1.15) của từng đồ thị.

Từ hình vẽ, có thể thấy thuật toán phổ thực hiện khá chính xác khi số lượng đỉnh của cộng đồng 1 phát hiện được xấp xỉ với số lượng thực tế. Độ tách biệt giữa hai cộng đồng có thể đánh giá qua hai giá trị quan trọng: phần tử lớn nhất nhỏ hơn 0 (x) và phần tử nhỏ nhất lớn hơn 0 (y). Khi có sự khác biệt rõ ràng giữa x và y, việc phân chia các cộng đồng sẽ dễ dàng hơn. Trong trường hợp các giá trị x và y gần nhau, việc phân loại các đỉnh sẽ trở nên khó khăn hơn.

Trong thí nghiệm 2: Các đồ thị ngẫu nhiên được tạo từ mô hình SBM với số đỉnh của hai cụm bằng nhau, mỗi cụm gồm 5000 đỉnh, và cấu trúc cộng đồng được thay đổi bằng cách điều chỉnh xác suất xuất hiện cạnh bên trong mỗi cộng đồng và giữa các cộng đồng.

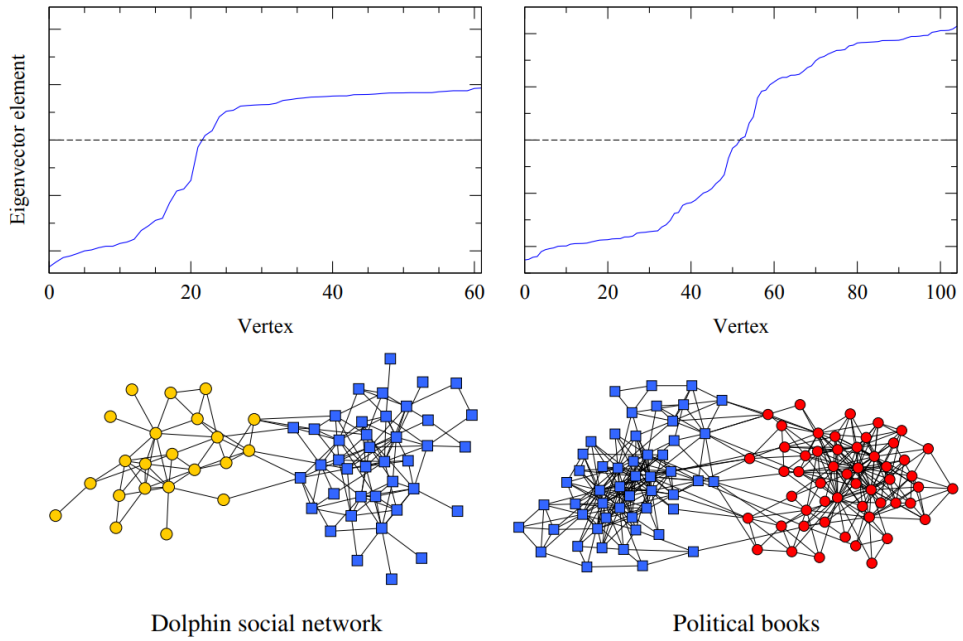
Trong Hình 3.2b, với các đồ thị có cấu trúc phân cụm rõ ràng, ta có thể thấy một đoạn thẳng vuông góc xuất hiện tại điểm giao giữa đường cong và

đường nét đứt. Điều này cho thấy các phần tử có giá trị khác 0 rõ rệt, giúp phân tách các cộng đồng một cách rành mạch và dễ dàng.

Ngược lại, trong trường hợp các đồ thị có cấu trúc phân cụm không rõ ràng, tại điểm giao chỉ có đường cong hoặc đường chéo mà không có đoạn thẳng rõ ràng. Điều này cho thấy các phần tử có giá trị gần bằng 0 của hai cộng đồng không tạo ra sự tách biệt, khiến việc phân loại các đỉnh trở nên khó khăn hơn và quá trình tách biệt cộng đồng kém hiệu quả.

Thí nghiệm trên đồ thị thực

Trong nghiên cứu của Newman [3], tác giả đã thực hiện các thí nghiệm trên hai mạng lưới thực tế: một mạng xã hội của cá heo [38] và một mạng về mối quan hệ giữa các cuốn sách chính trị. Kết quả của thí nghiệm trong bài báo đó được minh họa trong hình 3.3. Hình trên cho thấy biểu đồ vector riêng ứng với giá trị riêng lớn thứ hai, biểu diễn sự phân chia các đỉnh trong mạng lưới thành hai cộng đồng.



Hình 3.3: Thí nghiệm trên hai đồ thị thực về cá heo và sách chính trị.

Đối với mạng xã hội cá heo, biểu đồ vector riêng (bên trái) cho thấy sự phân chia rõ ràng thành hai cụm, tương ứng với các đỉnh trong đồ thị bên

dưới, nơi các đỉnh (tượng trưng cho cá heo) được tô màu vàng và xanh dương để chỉ ra hai cộng đồng khác nhau. Tương tự, đối với mạng lưới sách chính trị (bên phải), biểu đồ vector riêng cho thấy một sự phân chia thành hai cộng đồng, tương ứng với hai nhóm sách có khuynh hướng chính trị khác nhau, được thể hiện bằng màu xanh dương và màu đỏ. Phương pháp phổ đã cho thấy khả năng phân cụm các đỉnh trong các mạng phức tạp này một cách rõ ràng.

3.3.2. Thí nghiệm trên đồ thị có hướng

Các thí nghiệm này được chúng tôi thực hiện trên đồ thị có hướng, dựa trên thuật toán phương pháp phổ được đề xuất dành cho đồ thị có hướng, đã được trình bày ở phần 3.2.2.

Bên cạnh việc đánh giá thông qua các chỉ số modularity Q_u, Q_d, chúng tôi còn sử dụng chỉ số Jaccard để đo lường chất lượng phân cụm trên các đồ thị ngẫu nhiên có cấu trúc cộng đồng. Nếu phân cụm từ thuật toán trùng khớp hoàn toàn với phân cụm gốc của đồ thị thì chỉ số Jaccard bằng 1.

Thí nghiệm trên đồ thị ngẫu nhiên

Thí nghiệm trên mô hình phân vùng $l-$vùng:

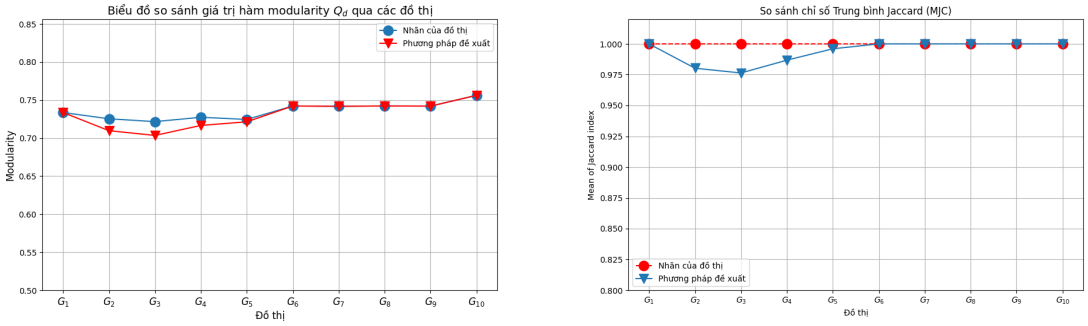
Trong thí nghiệm này, chúng tôi tạo ra 10 đồ thị ngẫu nhiên sử dụng mô hình $l-$vùng. Kích thước của các đồ thị được lựa chọn dao động từ vài trăm đến vài ngàn đỉnh. Mục đích của thí nghiệm là so sánh kết quả của thuật toán phương pháp phổ với nhãn cộng đồng ban đầu của các đồ thị được tạo ra.

Kết quả thí nghiệm được trình bày trong bảng 3.1. Bảng này bao gồm năm cột: Cột đầu tiên là tên của đồ thị, cột thứ hai là bộ tham số (g, l, p_{in}, p_{out}) tạo ra đồ thị theo mô hình $l-$vùng (đã được giới thiệu tại phần 1.6). Cột thứ ba là giá trị modularity Q_d từ các cộng đồng được phát hiện bằng thuật toán phổ, cột thứ tư là giá trị modularity Q_d dựa trên nhãn cộng đồng gốc của đồ thị, và cột cuối cùng là giá trị trung bình của chỉ số Jaccard (MJC) được giới thiệu tại phần 2.3.2, nhằm đánh giá mức độ tương đồng giữa các cộng đồng được phát hiện bởi thuật toán và nhãn gốc.

Đồ thị	$G = (g, l, p_{in}, p_{out})$	Q_d **từ kết quả thuật toán**	Q_d **theo nhãn**	**MJC**
G_1	(5,30,0.6,0.01)	0.734	0.734	1.000
G_2	(5,40,0.5,0.01)	0.710	0.725	0.980
G_3	(5,50,0.5,0.01)	0.704	0.722	0.976
G_4	(5,60,0.5,0.01)	0.717	0.727	0.987
G_5	(5,100,0.5,0.01)	0.721	0.724	0.996
G_6	(6,200,0.5,0.01)	0.742	0.742	1.000
G_7	(6,300,0.5,0.01)	0.742	0.742	1.000
G_8	(6,500,0.5,0.01)	0.742	0.742	1.000
G_9	(6,600,0.5,0.01)	0.742	0.742	1.000
G_{10}	(6,300,0.6,0.01)	0.756	0.756	1.000

Bảng 3.1: Kết quả thử nghiệm với mô hình phân vùng $l-$vùng.

Hình 3.4 minh họa trực quan hơn về kết quả này.



(a) Q_d (b) MJC

Hình 3.4: So sánh modularity Q_d và MJC của các đồ thị từ G_1 đến G_{10}.

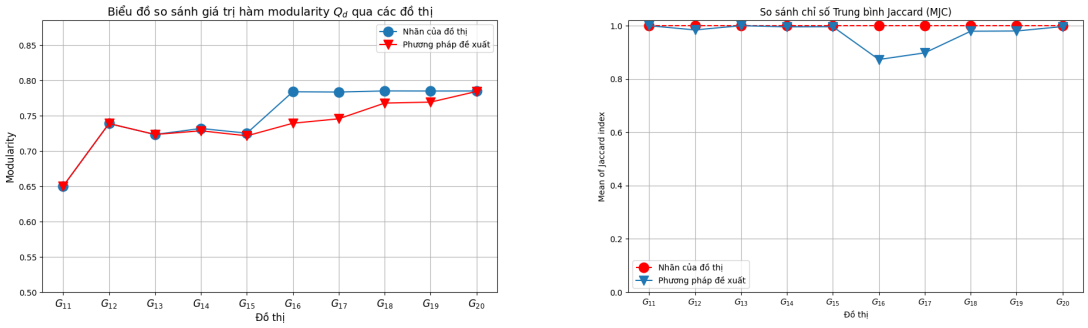
Thí nghiệm trên mô hình phân vùng Gaussian ngẫu nhiên: Chúng tôi cũng tạo ra 10 đồ thị ngẫu nhiên từ mô hình phân vùng Gaussian ngẫu nhiên. Kích thước của các đồ thị được lựa chọn dao động từ vài trăm tới vài ngàn đỉnh.

Kết quả thí nghiệm được trình bày trong bảng 3.2, được tổ chức tương tự bảng 3.1.

Kết quả được minh họa trực quan trong hình 3.5.

Đồ thị	$G = (n, \mu, \sigma, p_{in}, p_{out})$	Q_d **từ thuật toán**	Q_d **theo nhãn**	**MJC**
G_{11}	(100,30,4,0.7,0.01)	0.650	0.650	1.000
G_{12}	(200,40,5,0.7,0.01)	0.739	0.739	0.984
G_{13}	(400,80,10,0.7,0.01)	0.723	0.723	1.000
G_{14}	(500,100,10,0.7,0.01)	0.729	0.732	0.996
G_{15}	(500,100,10,0.6,0.01)	0.722	0.725	0.996
G_{16}	(1500,150,50,0.7,0.01)	0.739	0.784	0.873
G_{17}	(2000,200,50,0.7,0.01)	0.746	0.784	0.898
G_{18}	(2500,250,50,0.7,0.01)	0.768	0.785	0.980
G_{19}	(2500,250,50,0.7,0.01)	0.770	0.786	0.981
G_{20}	(3000,300,100,0.7,0.01)	0.785	0.785	0.997

Bảng 3.2: Kết quả thử nghiệm với mô hình phân vùng Gaussian.



(a) Q_d (b) MJC

Hình 3.5: So sánh modularity Q_d và MJC của các đồ thị từ G_{11} đến G_{20}.

Nhận xét 3.3.1. *Nhìn vào các bảng và hình ảnh trên, chúng ta thấy thuật toán phương pháp phổ đã mang lại chất lượng cộng đồng khá tốt trên cả hai mô hình l−vùng và Gaussian ngẫu nhiên. Sự chênh lệch giữa giá trị modularity Q_d từ thuật toán và nhãn cộng đồng ban đầu là không đáng kể (như minh họa trong hình 3.4a và 3.5a), trong khi chỉ số Jaccard trung bình (MJC) cho thấy độ tương đồng cao giữa các cộng đồng sinh ra từ thuật toán và nhãn ban đầu (hình 3.4b và 3.5b). Ta thấy có nhiều trường hợp kết quả từ thuật toán đem lại trùng khớp hoàn toàn với nhãn cộng đồng của đồ thị.*

Thí nghiệm trên đồ thị thực

Trong phần này, chúng tôi sẽ tiến hành thí nghiệm để so sánh phương pháp phổ cho đồ thị có hướng của chúng tôi với hai thuật toán phổ khác là

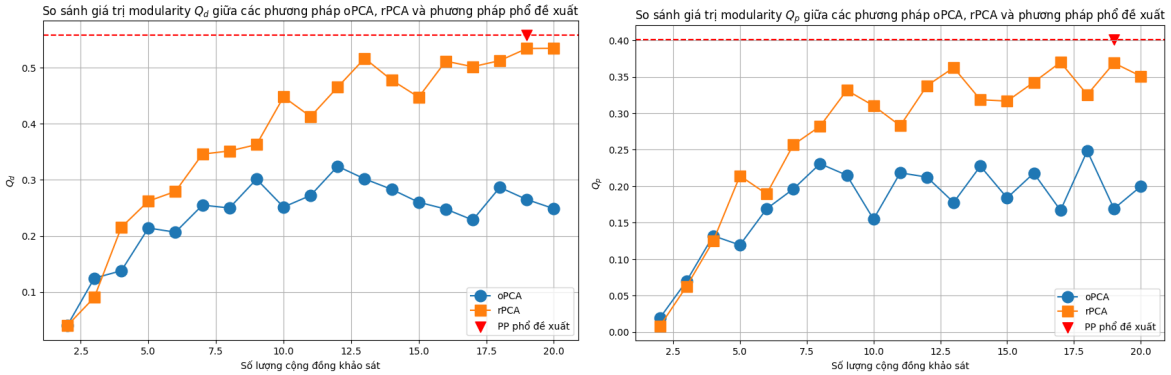
oPCA và rPCA [39]. Ý tưởng chính của hai thuật toán oPCA, rPCA chính là dựa trên việc phân rã các ma trận biểu diễn của đồ thị (lần lượt là ma trận kề A và ma trận Laplacian) thành tổng của các ma trận có hạng bằng 1, từ đó thu được vector biểu diễn của từng đỉnh. Các bộ dữ liệu mà chúng tôi sử dụng để so sánh được mô tả trong bảng 3.3.

Đồ thị đơn	$(\|V\|, \|E\|)$	Mô tả
Mạng Copenhagen [40]	$G_{r1} = (536, 924)$	Một mạng lưới tin nhắn giữa các sinh viên đại học trong Nghiên cứu Mạng lưới Copenhagen, trong khoảng thời gian bốn tuần
Mạng Copenhagen [40]	$G_{r2} = (568, 1303)$	Một mạng lưới cuộc gọi giữa các sinh viên đại học trong Nghiên cứu Mạng lưới Copenhagen, trong khoảng thời gian bốn tuần
Mạng Uni Email [41]	$G_{r4} = (1133, 10903)$	Một mạng lưới đại diện cho việc trao đổi email giữa các thành viên của Đại học Rovira i Virgili ở Tây Ban Nha, năm 2003.
Mạng Euroroad [42]	$G_{r3} = (1174, 1417)$	Một mạng lưới "E-roads" quốc tế, chủ yếu ở Châu Âu. Các đỉnh đại diện cho các thành phố và các cạnh đại diện cho các con đường.

Bảng 3.3: Danh sách các đồ thị đơn thực tế.

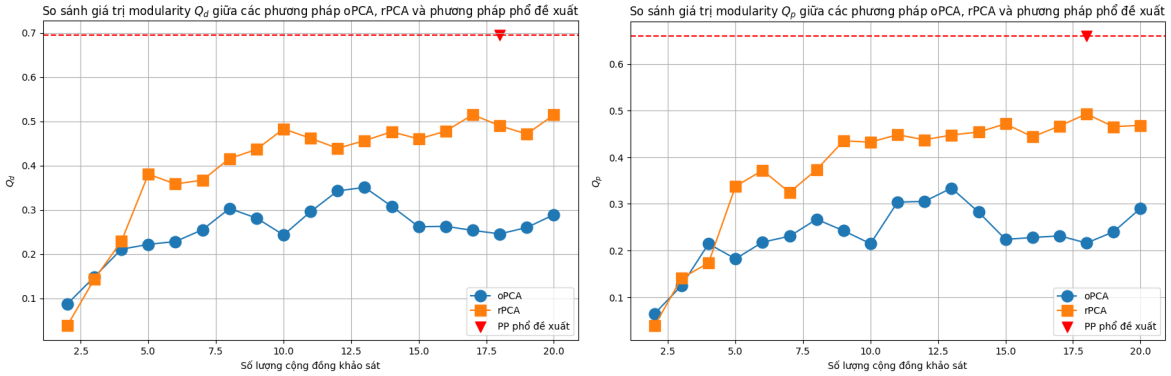
Vì các thuật toán oPCA và rPCA yêu cầu phải cung cấp trước số lượng cộng đồng, chúng tôi sẽ áp dụng hai thuật toán này với số cộng đồng thay đổi từ 2 đến 20 để tìm ra cách phân cụm tốt nhất (cách phân cụm có giá trị modularity lớn nhất). Kết quả này sẽ được so sánh với giá trị modularity của phân cụm thu được từ thuật toán của chúng tôi. Trong mỗi thí nghiệm, chúng tôi sử dụng cả hai chỉ số modularity là Q_d (xem công thức (1.4.4)) và Q_p (do chúng tôi đề xuất) nhằm đảm bảo việc đánh giá được khách quan hơn.

Kết quả thí nghiệm được tình bày tại bốn hình 3.6, 3.7, 3.8 và 3.9 dưới đây, trong đó trục hoành là số cộng đồng tương ứng với một cách phân cụm, trục tung là giá trị của hàm modularity, đường màu xanh với kí hiệu các đỉnh tròn ứng với kết quả thuật toán oPCA, đường màu cam với kí hiệu vuông ứng với thuật toán rPCA và đường màu đỏ nét đứt kí hiệu hình tam giác ứng với phương pháp phổ đề xuất.



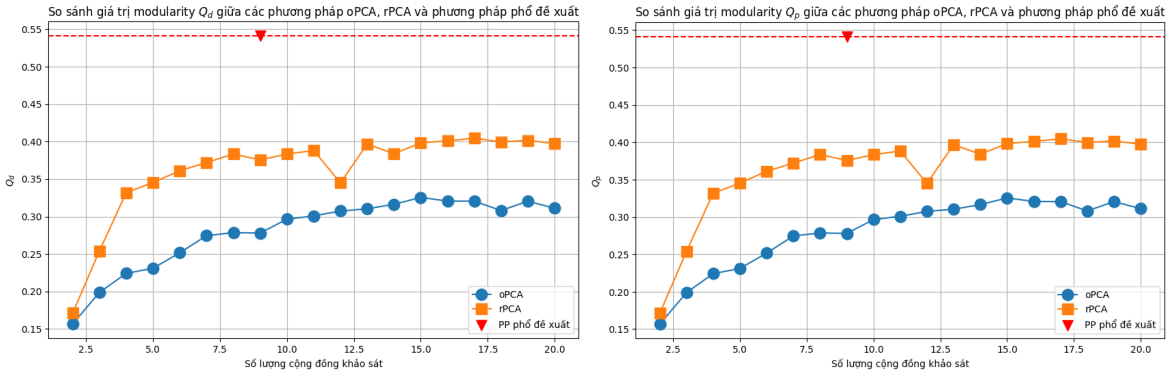
(a) Q_d (b) Q_p

Hình 3.6: So sánh các hàm modularity Q_d, Q_p giữa thuật toán phổ của chúng tôi và hai thuật toán phổ khác: oPCA, rPCA trên đồ thị G_{r1}.



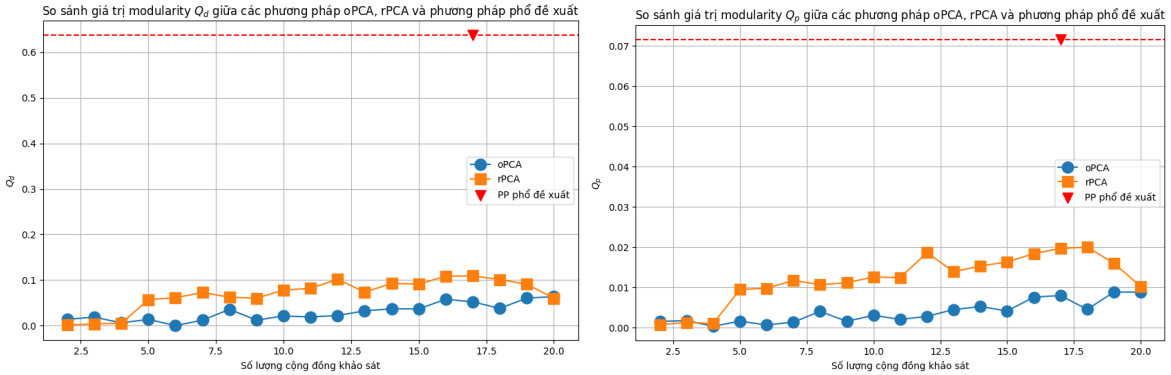
(a) Q_d (b) Q_p

Hình 3.7: So sánh các hàm modularity Q_d, Q_p giữa thuật toán phổ của chúng tôi và hai thuật toán phổ khác: oPCA, rPCA trên đồ thị G_{r2}.



(a) Q_d (b) Q_p

Hình 3.8: So sánh các hàm modularity Q_d, Q_p giữa thuật toán phổ của chúng tôi và hai thuật toán phổ khác: oPCA, rPCA trên đồ thị G_{r3}.



(a) Q_d (b) Q_p

Hình 3.9: So sánh các hàm modularity Q_d, Q_p giữa thuật toán phổ của chúng tôi và hai thuật toán phổ khác: oPCA, rPCA trên đồ thị G_{r4}.

Nhận xét 3.3.2. *Nhìn từ các hình 3.6, 3.7, 3.8 và 3.9, có thể rút ra những nhận xét sau:*

- *Thuật toán phổ đề xuất của chúng tôi (đường màu đỏ, ký hiệu hình tam giác) chỉ cần thực hiện một lần và trả về một điểm duy nhất trên biểu đồ, trong khi các thuật toán khác như oPCA và rPCA phải chạy nhiều lần với các số cụm khác nhau để tìm ra cách phân cụm tốt nhất.*
- *Giá trị modularity Q_d và Q_p của thuật toán phổ đề xuất đều đạt mức cao, cho thấy chất lượng phân cụm rất tốt.*
- *So sánh với oPCA và rPCA, phương pháp phổ của chúng tôi thể hiện hiệu suất vượt trội.*

So sánh với thuật toán mở rộng của Louvain trên đồ thị có hướng

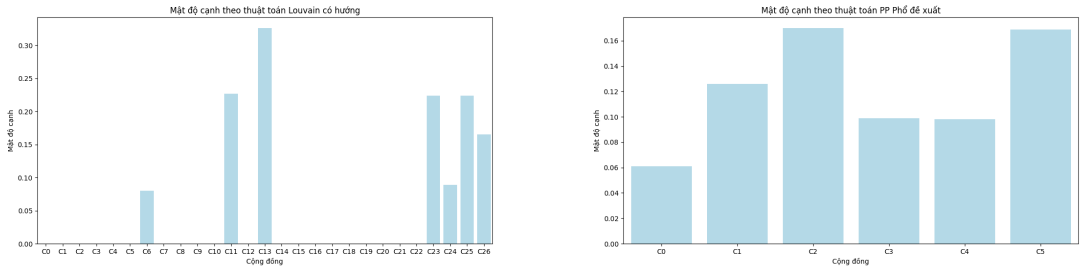
Trong phần này, luận văn sẽ trình bày việc so sánh phương pháp phổ đề xuất và thuật toán Louvain có hướng (Directed Louvain [19]) trên đồ thị thực EU Email ($G_{r5} = (1005, 25571)$) là đồ thị email bao gồm email đi và đến của một tổ chức nghiên cứu ở Châu Âu.

Nhận xét 3.3.3. *Kết quả giá trị moudalrity Q_d từ phương pháp đề xuất là* 0.361 *trong khi giá trị đó của thuật toán Louvain có hướng là* 0.433.

Tuy nhiên, khi đánh giá sâu hơn về sự liên kết, tính chất cộng đồng thì thuật toán Louvain có hướng có tới hơn 70% các cộng đồng có tính chất kém khi số lượng đỉnh

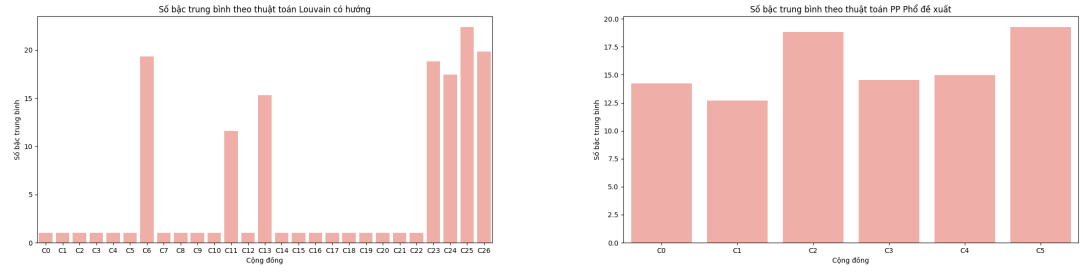
chỉ là 1 đỉnh dẫn tới số cạnh của các cộng đồng đó rất ít. Chi tiết được minh hoạ qua các hình 3.10 và 3.11, cụ thể:

- *Hình 3.10b, 3.11b thể hiện mỗi cộng đồng được phát hiện từ thuật toán đề xuất có mật độ cạnh và số bậc trung bình là đủ lớn, tương đồng với các cộng đồng khác.*
- *Hình 3.10a, 3.11a thể hiện chỉ có 7/27 cộng được phát hiện được từ thuật toán Louvain có hướng là có mật độ cạnh và số bậc trung bình đủ cao để thấy đó là một cộng đồng có liên kết chặt chẽ.*



(a) Thuật toán Louvain có hướng (b) Phương pháp phổ đề xuất

Hình 3.10: Giá trị mật độ cạnh của các cộng đồng phát hiện được.



(a) Thuật toán Louvain có hướng (b) Phương pháp phổ đề xuất

Hình 3.11: Số bậc trung bình của các đỉnh trong các cộng đồng phát hiện được.

Nhận xét 3.3.4. *Dựa trên nhận xét trên cùng các hình kết quả, ta thấy:*

- *Bởi vì thuật toán Louvain hay thuật toán Louvain có hướng chỉ có tiêu chí duy nhất để di chuyển đỉnh là hàm modularity, mà hàm modularity chỉ phản ánh một góc nhìn về cộng đồng "tốt", do đó sẽ có một số tính chất khá trực quan về đồ thị mà hàm modulariy chưa thể bao quát, ví dụ như số bậc trung bình,*

số đỉnh trong cụm hay chính sự kết nối giữa các cộng đồng con (mà thuật toán Leiden cho đồ thị vô hướng đã giải quyết).

- *Trong khi đó, thuật toán dựa trên phương pháp phổ thì tiến hành phân tích phổ của ma trận biểu diễn, mà tính chất của đồ thị thì có thể được tích hợp thông qua các giá trị riêng, vector riêng, do đó các cụm thu được có sự thống nhất về tính chất hơn.*

Kết luận

Luận văn "Một số thuật toán tìm kiếm cộng đồng mạng thông qua tối ưu hoá hàm modularity" đã trình bày về bài toán tìm kiếm cộng đồng mạng và một số thuật toán sử dụng tối ưu hoá modularity để tìm kiếm cộng đồng mạng. Cụ thể, luận văn đã:

- Trình bày thuật toán Louvain và thuật toán cải thiện của nó là thuật toán Leiden.
- Trình bày phương pháp phổ của Newman, dựa trên phân tích giá trị riêng và vector riêng của ma trận Laplacian.
- Trình bày định nghĩa hàm modularity mới cho đồ thị có hướng phù hợp với định nghĩa hàm modularity trong trường hợp vô hướng sử dụng bước đi ngẫu nhiên và phân phối dừng.
- Mở rộng phương pháp phổ cho đồ thị có hướng, sử dụng modularity mà chúng tôi đã đề xuất.
- Thực hiện các thí nghiệm trên dữ liệu sinh ngẫu nhiên và dữ liệu thực để chứng tỏ tính hợp lý và hiệu quả của phương pháp phổ mà chúng tôi đã đề xuất.

Việc xây dựng các hàm modularity mới sao cho phù hợp với từng loại đồ thị cụ thể luôn là một chủ đề hấp dẫn. Trong thời gian tới, chúng tôi sẽ tiếp tục nghiên cứu để xây dựng các hàm modularity mới cũng như xây dựng các thuật toán tìm kiếm cộng đồng cho đồ thị hai phần.

Danh mục công trình của tác giả

- Tien Dat Dang, Duy Hieu Do, and Thi Ha Duong Phan. Community detection in directed graphs using stationary distribution and hitting times methods. Social Network Analysis and Mining, 13:1–30, 2023.

Tài liệu tham khảo

[1] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P10008, 2008.

[2] Vincent Antonio Traag, Ludo Waltman, and Nees Jan van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9, 2018.

[3] Mark E. J. Newman. Spectral methods for network community detection and graph partitioning. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 88 4:042822, 2013.

[4] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, September 2000.

[5] Mark Newman. *Networks: An Introduction*. Oxford University Press, 03 2010.

[6] Ciriaco Valdez-Flores Richard M. Feldman. *Applied Probability and Stochastic Processes*. Springer, 2010.

[7] Santo Fortunato. Community detection in graphs. *ArXiv*, abs/0906.0612, 2009.

[8] Songlai Ning, Jiakang Li, and Y. Lu. A comprehensive review of community detection in graphs. *ArXiv*, abs/2309.11798, 2023.

[9] Mark Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 69 2 Pt 2:026113, 2003.

[10] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[11] J Chitra Devi and E. Poovammal. An analysis of overlapping community detection algorithms in social networks. *Procedia Computer Science*, 89:349–358, 2016.

[12] Elizabeth A. Leicht and Mark E. J. Newman. Community structure in directed networks. *Physical review letters*, 100 11:118703, 2007.

[13] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[14] Jiyang Chen, Osmar R Zaiane, and Randy Goebel. Detecting communities in social networks using max-min modularity. In *SDM*, 2009.

[15] Rumi Ghosh and Kristina Lerman. Community detection using a measure of global influence. In *Social Network Mining and Analysis*, 2008.

[16] Zhidan Feng, Xiaowei Xu, Nurcan Yuruk, and Thomas A. J. Schweiger. A novel similarity-based modularity function for graph partitioning. In *International Conference on Data Warehousing and Knowledge Discovery*, 2007.

[17] Nguyễn Thị Bạch Kim. Giáo trình các phương pháp tối ưu, lý thuyết và thuật toán. 2008.

[18] Jicun Zhang, Jiyou Fei, Xueping Song, and Jiawei Feng. An improved louvain algorithm for community detection. *Mathematical Problems in Engineering*, 2021.

[19] Nicolas Dugué and Anthony Perez. Directed louvain: maximizing modularity in directed networks. 2015.

[20] Santo Fortunato and Andrea Lancichinetti. Community detection algorithms: a comparative analysis: invited presentation, extended abstract. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 80 5 Pt 2:056117, 2009.

[21] Pascal Pons and Matthieu Latapy. Journal of graph algorithms and applications computing communities in large networks using random walks.

[22] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks: [extended abstract]. *ArXiv*, abs/cs/0702048, 2007.

[23] Konstantin Avrachenkov, Aleksei Yu. Kondratev, and Vladimir V. Mazalov. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452 – 473, 1977.

[24] http://www.cs.cornell.edu/projects/kddcup/

[25] Réka Albert, Hawoong Jeong, and Albert Ĺaszló Barabási. Internet: Diameter of the world-wide web. *Nature*, 401:130–131, 1999.

[26] Renaud Lambiotte, Vincent D. Blondel, Cristobald de Kerchove, Etienne Huens, Christophe Prieur, Zbigniew Smoreda, and Paul Van Dooren. Geographical dispersal of mobile communication networks. *Physica A-statistical Mechanics and Its Applications*, 387:5317–5325, 2008.

[27] http://law.dsi.unimi.it/ (Laboratory for Web Algorithmics).

[28] http://dbpubs.stanford.edu:8091/testbed/doc2/WebBase/ (Stanford WebBase Project).

[29] https://snap.stanford.edu/data/.

[30] https://sparse.tamu.edu/Barabasi/NotreDame_actors.

[31] http://law.di.unimi.it/webdata/uk-2005/.

[32] Tien Dat Dang, Duy Hieu Do, and Thi Ha Duong Phan. Community detection in directed graphs using stationary distribution and hitting times methods. *Social Network Analysis and Mining*, 13:1–30, 2023.

[33] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert G¨orke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. Maximizing modularity is hard. *arXiv: Data Analysis, Statistics and Probability*, 2006.

[34] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms, 2nd edition. 2001.

[35] Pascal Pons and Matthieu Latapy. Journal of graph algorithms and applications computing communities in large networks using random walks.

[36] Michael B. Cohen, Jonathan A. Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. Faster algorithms for computing the stationary distribution, simulating random walks, and more. *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 583–592, 2016.

[37] Elchanan Mossel, Joe Neeman, and Allan Sly. Stochastic block models and reconstruction. *arXiv: Probability*, 2012.

[38] David Lusseau, Karsten Schneider, Oliver Boisseau, Patti A. Haase, Elisabeth Slooten, and Stephen Michael Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54:396–405, 2003.

[39] Zhe Wang, Yingbin Liang, and Pengsheng Ji. Spectral algorithms for community detection in directed networks. *ArXiv*, abs/2008.03820, 2020.

[40] P. Sapiezynski, et al., "Interaction data from the Copenhagen Networks Study." Scientific Data 6, 315 (2019), https://doi.org/10.1038/s41597-019-0325-x.

[41] R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt and A. Arenas, "Self-similar community structure in a network of human interactions." Physical Review E 68, 065103(R) (2003)., https://doi.org/10.1103/physreve.68.065103.

[42] L. Šubelj and M. Bajec. "Robust network community detection using balanced propagation." The European Physical Journal B 81(3), 353-362 (2011)., https://doi.org/10.1140/epjb/e2011-10979-2.