BỘ GIÁO DỤC VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Nguyễn Hương Quỳnh

TÌM HIỂU VỀ CÁC THUẬT TOÁN PHÂN CỤM CHO CÁC ĐỒ THỊ LỚN HAI PHẦN

LUẬN VĂN THẠC SĨ TOÁN ỨNG DỤNG

Hà Nội - 2023

BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Nguyễn Hương Quỳnh

TÌM HIỂU VỀ CÁC THUẬT TOÁN PHÂN CỤM CHO CÁC ĐỒ THỊ LỚN HAI PHẦN

LUẬN VĂN THẠC SĨ TOÁN ỨNG DỤNG

Mã số: 8460112

NGƯỜI HƯỚNG DẪN KHOA HỌC:

PGS.TSKH Phan Thị Hà Dương

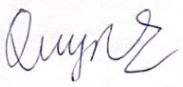
Hà Nội – 2023

LỜI CAM ĐOAN

Tôi xin cam đoan luận văn này là sự tìm tòi, học hỏi, trau dồi kiến thức của bản thân dưới sự hướng dẫn tận tình của cô Phan Thị Hà Dương. Mọi kết quả nghiên cứu cũng như ý tưởng của tác giả khác, nếu có đều được trích dẫn cụ thể. Tôi xin chịu trách nhiệm về những lời cam đoan.

Hà Nội, tháng 10 năm 2023

Học viên



Nguyễn Hương Quỳnh

LỜI CẢM ƠN

Đầu tiên, tôi xin được tỏ lòng biết ơn sâu sắc nhất của mình tới PGS.TSKH. Phan Thị Hà Dương, cô đã trực tiếp hướng dẫn và giúp đỡ tôi tìm ra đề tài luận văn cũng như định hình hướng nghiên cứu. Không chỉ là người hướng dẫn khoa học tận tâm, cô còn cho tôi những lời khuyên, động viên, khích lệ giúp tôi trưởng thành hơn trong cuộc sống.

Tôi xin chân thành cảm ơn TS. Đỗ Duy Hiếu - Viện Toán học đã hướng dẫn, góp ý và giúp đỡ tôi rất nhiều trong quá trình tôi đọc tài liệu và làm luận văn.

Tôi xin chân thành cảm ơn nhóm seminar đã đồng hành cùng trong việc tìm hiểu về kiến thức liên quan đến bài toán phân cụm cho đồ thị và góp ý cho tôi trong quá trình hoàn thành luận văn.

Tôi xin chân thành cảm ơn sự hỗ trợ của Công ty Cổ phần Tập đoàn Vingroup đã tài trợ cho quá trình học tập của tôi trong hai năm qua. Tôi được hỗ trợ bởi Chương trình học bổng Thạc sĩ, Tiến sĩ của Quỹ đổi mới sáng tạo Vingroup (VINIF), Viện Dữ liệu lớn, mã số VINIF.2021.ThS.VTH.02 và VINIF.2022.ThS.076.

Trong thời gian học tập tại Viện Toán học, tôi đã nhận được nhiều sự quan tâm, góp ý, hỗ trợ quý báu của các thầy cô, anh chị và bạn bè. Tôi xin được chân thành chân thành bày tỏ lòng biết ơn sâu sắc đến các thầy cô, anh chị và bạn bè.

Tôi cũng xin trân trọng cảm ơn Viện Toán học và cơ sở đào tạo là Học viện Khoa học và Công nghệ, Viện Hàn lâm Khoa học và Công nghệ Việt Nam đã giúp đỡ và tạo điều kiện thuận lợi về môi trường học tập cho tôi trong suốt quá trình thực hiện Luận văn này.

Cuối cùng, tôi xin tỏ lòng biết ơn vô hạn tới gia đình tôi, bố mẹ luôn kiên nhẫn và thương yêu tôi vô điều kiện.

Mục lục

LỜI MỞ ĐẦU

Tổng quan tình hình nghiên cứu và sự cần thiết tiến hành nghiên cứu

Bài toán phát hiện cộng đồng xuất hiện và phát triển vào những năm 80 của thế kỉ trước.

Đến nay các nhà khoa học đã tìm ra được một số thuật toán để giải quyết bài toán phân cụm như: thuật toán Louvain, thuật toán sử dụng bước đi ngẫu nhiên, thuật toán dựa trên động lực học khoảng cách.

Đã có rất nhiều nhà khoa học nghiên cứu về bài toán này và có nhiều bài báo được đăng trên các tạp chí uy tín.

Trong khi không gian mạng ngày càng lớn và ngày càng phát triển, bài toán phát hiện cộng đồng hay phân cụm đồ thị càng được quan tâm và mở rộng hơn nữa. Bài toán phát hiện cộng đồng trong thực tế là nhóm những đỉnh có nhiều điểm chung, nhiều liên kết lại với nhau trong toán học là nhóm những đỉnh có mật độ liên kết lớn so với số đỉnh của đồ thị. Ví dụ đồ thị mạng lưới bạn bè trên facebook hai người kết bạn với nhau thì có một cạnh nối giữa hai người, việc phân cụm những nhóm bạn có nhiều bạn chung, có nhiều tương tác với nhau có thể giúp facebook đề xuất một số gợi ý kết bạn cho người dùng. Bài toán phát hiện cộng đồng cho đồ thị hai phần cũng xuất hiện nhiều trong thực tế. Ví dụ bài toán phân cụm đồ thị hai phần sau: phần 1 các đỉnh của đồ thị là các nhà khoa học và phần 2 các đỉnh của đồ thị các bài báo các nhà khoa học viết. Các cạnh của đồ thị được nối là các nhà khoa học với bài báo họ tham gia viết. Bài toán đặt ra là phân cụm các nhà khoa học có chung nhiều bài báo vào cùng một cụm.

Hiện nay bài toán phát hiện cộng đồng trên đồ thị hai phần (một lớp đồ thị quan trọng) nhưng chưa có nhiều kết quả (theo hiểu biết của chúng tôi) nên bài toán này cần được nghiên cứu nhiều hơn nữa.

Mục đích, đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: đồ thị lớn, đồ thị lớn hai phần.

Phạm vi nghiên cứu: Định nghĩa tính chất về đồ thị, đồ thị lớn và đồ thị hai phần, bài toán phân cụm cho đồ thị lớn và đồ thị lớn hai phần.

Phương pháp nghiên cứu

Đọc hiểu và trình bày hệ thống các kiến thức liên quan đến đề tài luận văn từ các tài liệu tham khảo chuyên ngành.

Cấu trúc và dự kiến kết quả đạt được của luận văn

Ngoài phần Lời mở đầu, Lời cảm ơn, Lời cam đoan, Kết luận và Tài liệu tham khảo, Luận văn được chia thành bốn chương.

- Chương 1: Kiến thức chuẩn bị.
- Chương 2: Thuật toán phân cụm trên đồ thị lớn.
- Chương 3: Thuật toán phân cụm trên đồ thị lớn hai phần.
- Chương 4: Một số thí nghiệm.

Chương 1
KIẾN THỨC CHUẨN BỊ

1.1. Một số khái niệm về lý thuyết đồ thị

Định nghĩa 1.1. Đồ thị vô hướng

Đồ thị vô hướng G là một cặp có thứ tự $G = (V, E)$, ở đây V là một tập hợp các đỉnh hoặc nút, còn E là tập các cặp không có thứ tự chứa các đỉnh phân biệt, được gọi là cạnh. Hai đỉnh thuộc một cạnh được gọi là các đỉnh đầu cuối của cạnh đó.

Định nghĩa 1.2. Đồ thị có hướng

Đồ thị có hướng G là một cặp có thứ tự $G = (V, E)$, ở đây V là một tập hợp các đỉnh hoặc nút, còn E là tập các cặp có thứ tự chứa các đỉnh phân biệt, được gọi là cạnh. Cụ thể hơn nếu $(a, b) \in E$ thì (a, b) là cạnh của G với đỉnh đầu là a, đỉnh cuối là b và có hướng đi từ a đến b.

Định nghĩa 1.3. Đồ thị hai phần

Một đồ thị hai phần xác định bởi: $G = (U, V, E)$ trong đó $U \cup V$ là tập đỉnh của đồ thị; U, V rời nhau. U là tập hợp các đỉnh trái, V là các đỉnh phải và không có cạnh nối hai đỉnh bất kỳ thuộc cùng một tập hay $E \subseteq U \times V$ là tập hợp các cạnh liên kết giữa U và V.

Định nghĩa 1.4. Hành trình; chu trình

$G(V, E)$ là một đồ thị vô hướng. Một hành trình trong G là một dãy các đỉnh $v_0v_1v_2...v_n$ sao cho với mọi $i = 0, 1, ..., n-1, \{v_i, v_{i+1}\}$ là một cạnh của G. Các cạnh $\{v_i, v_{i+1}\}, i = 0, 1, ..., n-1$, cũng được gọi là các cạnh của hành trình $v_0v_1...v_n$. Khi đó n được gọi là độ dài, v_0 được gọi là đỉnh đầu, v_n được gọi là đỉnh cuối của hành trình trên. Một hành trình được gọi là khép kín nếu đỉnh đầu và đỉnh cuối của nó trùng nhau.

Một hành trình được gọi là đường nếu các đỉnh của hành trình đó đều khác nhau.

Một hành trình khép kín được gọi là chu trình, nếu nó có độ dài ít nhất là 3 và khi xoá

đi đỉnh cuối thì trờ thành đường.

1.2. Khoa học mạng và cộng đồng mạng

1.2.1. Khoa học mạng

Khoa học mạng đã phát triển thành một ngành học khổng lồ. Trong phần này, chúng tôi sẽ giải thích một số kiến thức cơ bản về khoa học mạng. Hiện nay có một số sách giáo trình tốt về khoa học mạng; trong số đó, chúng tôi đề cập đến tài liệu [1] với phạm vi bao phủ rộng và [2] tập trung duy nhất vào mô hình hóa, giải thích và chất lượng dữ liệu.

Có nhiều định nghĩa khoa học mạng và ta có thể tìm được một định nghĩa về khoa học mạng trong [3]: Khoa học mạng là nghiên cứu về các mạng sử dụng lý thuyết toán học, tập trung vào phân tích và mô tả đặc điểm và trạng thái của mạng. Nghiên cứu về các mạng đã thấy những nghiên cứu quan trọng, với trọng tâm là tìm hiểu và đánh giá các đặc tính thống kê của các mạng quy mô lớn (Newman, 2003).

1.2.2. Cộng đồng

Trong mục này chúng tôi chủ yếu tham khảo phần trình bày về cộng đồng ở trong tài liệu [4] và [5].

CẤU TRÚC CỘNG ĐỒNG

Các mạng thường được biểu diễn bằng các đồ thị trong đó một nhóm các nút (đỉnh) có các liên kết giữa chúng (các cạnh) [4]. Lý thuyết đồ thị là toán học của các mạng đang được sử dụng để mô hình hóa đồ thị (Stam, 2014). Erdös và Rényi (1959) đã giới thiệu các đồ thị ngẫu nhiên trong đó xác suất cạnh giữa hai đỉnh là như nhau đối với bất kỳ cặp nào khác. Nhưng các mạng trong thế giới thực không phải là đồ thị ngẫu nhiên vì chúng có một trật tự tốt của các mẫu. Một trong những đặc điểm thích hợp của các mạng trong thế giới thực là chúng có cấu trúc cộng đồng, có thể được mô hình hóa một cách tương đối chính xác bằng cách sử dụng đồ thị (Fortunato, 2010; Singh, 2014). Nói chung, cộng đồng hoặc cụm được định nghĩa là một nhóm các đỉnh có các liên kết chặt chẽ khác với phần còn lại của mạng (Yang và cộng sự, 2010). Xác định cấu trúc cộng đồng là một bước tiến tới sự hiểu biết về các cấu trúc mạng khác nhau (Newman và Girvan, 2004) với các ứng dụng trong một số lĩnh vực như mạng xã hội trực tuyến và tất cả các ngành khoa học vật lý và đời sống (Lewis, 2011).

Phát hiện cộng đồng đề cập đến việc xác định các nhóm đỉnh có nhiều liên kết trong mạng tùy thuộc vào thuộc tính cấu trúc của chúng (Yang và cộng sự, 2013; Kelley và cộng

sự, 2012). Nhiều thuật toán để phát hiện cộng đồng đã được phát triển, sử dụng các kỹ thuật và công cụ từ các ngành khác nhau như sinh học, vật lý, khoa học xã hội, toán học ứng dụng và khoa học máy tính (Lancichinetti và Fortunato, 2009). Tuy nhiên, một thuật toán phát hiện cộng đồng duy nhất không hoạt động trong tất cả các loại mạng (Plantié và Crampes, 2013; Yang và cộng sự, 2016) vì có rất nhiều mạng phức tạp được tạo từ các cách khác nhau [4]. Vì vậy bài toán phát hiện cộng đồng vẫn đang được quan tâm và ngày càng có nhiều thuật toán phát hiện cồng đồng được đề xuất.

Nhiều hệ thống phức tạp trong tự nhiên và xã hội có thể được mô tả dưới dạng đồ thị (mạng lưới) giới hạn mạng lưới kết nối phức tạp giữa các đơn vị mà chúng được tạo thành [4]. Đồ thị đã nổi lên như một mô hình mạnh mẽ để biểu diễn các loại dữ liệu khác nhau. Ví dụ: dữ liệu không có cấu trúc (ví dụ: tài liệu văn bản), dữ liệu bán cấu trúc (ví dụ: cơ sở dữ liệu XML) và dữ liệu có cấu trúc (ví dụ: cơ sở dữ liệu quan hệ) đều có thể được mô hình hóa dưới dạng đồ thị, trong đó các đỉnh (nút) tương ứng và các cạnh, tương ứng, có thể là siêu liên kết, mối quan hệ cha-con và mối quan hệ chính-khóa ngoại. Ngoài ra, các biểu đồ tự nhiên phát sinh trong các lĩnh vực ứng dụng như mạng sinh học, mạng xã hội mạng thông tin.

Cấu trúc cộng đồng tồn tại một cách tự nhiên trong nhiều mạng lưới trong thế giới thực như mạng xã hội, mạng sinh học, mạng cộng tác và truyền thông chỉ là một vài ví dụ [4]. Một câu hỏi được quan tâm là làm thế nào để giải thích cấu trúc của các mạng như vậy về sự cùng tồn tại của các tiểu đơn vị cấu trúc của chúng (cộng đồng) gắn với các bộ phận có liên kết với chúng cao hơn. Việc xác định các cộng đồng chưa biết trước này (ví dụ, các nhóm người, các ngành công nghiệp và các protein liên quan đến chức năng) là rất quan trọng đối với sự hiểu biết về các đặc tính cấu trúc và chức năng của các mạng.

Để minh họa, chúng tôi giới thiệu ba ví dụ đáng chú ý về các cộng đồng [5].

- Mạng xã hội trực tuyến. Một loại mạng mà cộng đồng thường xuyên được quan sát là mạng xã hội trực tuyến. Được kích hoạt bởi Internet và được thúc đẩy bởi sự ra đời gần đây của các trang mạng xã hội trực tuyến như Facebook, Google+ và Twitter, nghiên cứu về khám phá cộng đồng trên các mạng xã hội trực tuyến đã và đang bùng nổ. Với sự sẵn có của dữ liệu mạng xã hội quy mô lớn, nghiên cứu đã dẫn đến sự phát triển của nhiều ứng dụng thú vị, ví dụ: khám phá vòng kết nối xã hội và tìm kiếm cộng đồng có ảnh hưởng.

- Ngoài ra, do sự phát triển của các thiết bị điện thoại thông minh, các mạng xã hội

trực tuyến đã dẫn đến sự phát triển nhanh chóng của các mạng xã hội địa lý (còn được gọi là mạng xã hội dựa trên vị trí), chẳng hạn như Foursquare, Yelp, Google+ và Facebook Places. Trong mạng xã hội địa lý, người dùng được liên kết với thông tin vị trí (ví dụ: quê quán và địa điểm đăng ký) và cộng đồng bao gồm những người dùng được kết nối chặt chẽ trong lớp xã hội cũng như gần nhau về mặt không gian trong lớp không gian.

- Mạng cộng tác học thuật. Mạng cộng tác học thuật nổi tiếng là mạng DBLP, trong đó một đỉnh đại diện cho một tác giả và một cạnh giữa hai tác giả biểu thị mối quan hệ cộng tác, tức là họ có (các) ấn phẩm đồng tác giả. Ngoài ra, các đỉnh có thể có các thuộc tính đại diện cho lĩnh vực chuyên môn của tác giả như ấn phẩm, bài báo, sách. Các cộng đồng trong mạng DBLP có thể đại diện cho một nhóm tác giả thường xuyên cộng tác với nhau và làm việc về các chủ đề tương tự.

Mạng cộng tác học thuật trong ví dụ 3 là một mạng thông tin không đồng nhất [5]. Mạng không đồng nhất bao gồm các đỉnh và các cạnh của cả hai loại khác nhau. Ví dụ, trong một mạng lưới chăm sóc sức khỏe, các đỉnh có thể là bệnh nhân, bác sĩ, xét nghiệm y tế, bệnh tật, thuốc men, bệnh viện, phương pháp điều trị,... Một mặt, việc coi tất cả các đỉnh cùng loại có thể bỏ sót thông tin quan trọng. Mặt khác, việc coi mọi đỉnh là một kiểu riêng biệt có thể bỏ sót bức tranh lớn. Đây là một ví dụ cổ điển về một mạng không đồng nhất. Nhiều loại đối tượng như vậy, các mạng thông tin liên kết với nhau, không đồng nhất nhưng thường là bán cấu trúc, làm cho các cộng đồng trên các mạng thông tin không đồng nhất trở nên phức tạp và thú vị.

Trong luận văn này ngoài việc giải quyết các bài toán phát hiện cộng đồng thông thường có tất cả các đỉnh cùng loại (chương 2) thì luận văn còn tìm hiểu về cách phát hiện cộng đồng trên các đồ thị hai phần để giải quyết cho các mạng mà có hai loại đỉnh khác nhau (chương 3). Ngoài ví dụ về mạng các tác giả và các sản phẩm của họ thì trong thực tế ta cũng bắt gặp rất nhiều các mạng gồm hai loại đỉnh khác nhau chẳng hạn như:

- Đồ thị biểu diễn một số diễn viên và một số phim họ tham gia đóng. Khi đó tập các diễn viên và tập các bộ phim được liên kết bởi các cạnh nếu diễn viên tham gia đóng bộ phim nào thì sẽ có cạnh nối giữa chúng.

- Đồ thị biểu diễn các công ty sản xuất và người tiêu dùng. Trong đó tập đỉnh gồm hai tập con là công ty sản xuất và người tiêu dùng. Nếu người tiêu dùng sử dụng sản phẩm của công ty thì sẽ có cạnh liên kết giữa chúng.

PHÁT HIỆN CỘNG ĐỒNG

Phát hiện cộng đồng trên mạng xã hội cũng là một nhiệm vụ rất quan trọng trong phân tích mạng xã hội. Do tầm quan trọng của các cộng đồng mạng xã hội và khả năng ứng dụng to lớn của chúng trong các lĩnh vực khác nhau đã có nhiều các thuật toán phát hiện cộng đồng trên mạng xã hội đã được đề xuất. Tuy nhiên, hầu hết các thuật toán chưa đạt được hiệu quả trong việc phát hiện cộng đồng trên các mạng xã hội quy mô rất lớn hiện nay. Đồng thời, cùng với sự phát triển mạnh mẽ của công nghệ thông tin thì việc sử dụng các mạng xã hội của chúng ta đang phát triển theo cấp số nhân và hệ quả là quy mô của mạng xã hội phát triển nhanh chóng và trở nên khổng lồ. Điều này dẫn đến việc phát hiện cộng đồng trên các mạng xã hội quy mô rất lớn không thể giải quyết bằng các thuật toán truyền thống do độ phức tạp về thời gian và không gian tính toán. Có nghĩa là, hầu hết các thuật toán hiện có không thể được mở rộng đến kích thước khổng lồ của các mạng xã hội. Để giải quyết được thách thức đặt ra, cần đề xuất các phương pháp giảm kích thước của mạng xã hội để thực hiện phát hiện cộng đồng mạng xã hội hiệu quả đồng thời vẫn phải đảm bảo được các tính chất của cộng đồng mạng xã hội ban đầu là rất ý nghĩa, cần thiết và quan trọng [4].

Mục tiêu của bài toán phát hiện cộng đồng mạng xã hội là từ các mạng xã hội cho trước, phát hiện được các cộng đồng nằm trong đó và tìm hiểu về mối liên hệ bên trong các cộng đồng cũng như giữa các cộng đồng với nhau, mối liên hệ đó có ảnh hưởng thế nào đến toàn mạng xã hội [4]. Một tập hợp các đỉnh trên đồ thị được coi là một cộng đồng nếu mật độ cạnh giữa các đỉnh bên trong nó cao hơn so với mật độ của các cạnh giữa đỉnh của nó và những đỉnh khác bên ngoài. Phát hiện cộng đồng nhằm mục đích nhóm các đỉnh liên kết mạnh theo các mối quan hệ giữa chúng để tạo thành các đồ thị con từ đồ thị ban đầu. Các mạng xã hội thường được biểu diễn dưới dạng đồ thị đơn nên việc phát hiện cộng đồng trên mạng xã hội dựa trên cơ sở lý thuyết đồ thị còn được gọi là bài toán phân cụm đồ thị.

Bài toán: Phát hiện các cộng đồng trong mạng xã hội.

Đầu vào: Đồ thị mạng xã hội $G = (V, E)$ gồm tập V có các đỉnh: $v_1, v_2, \ldots, v_n$ và tập E các cạnh $E = \{(v_i, v_j)\}$.

Đầu ra: Tập các cộng đồng mạng xã hội C.

THUẬT TOÁN PHÁT HIỆN CỘNG ĐỒNG

Trong nhiều thập kỷ qua, số lượng các giải pháp phát hiện cộng đồng trên mạng xã hội đã được nghiên cứu là rất nhiều, thường xuyên và liên tục. Hiện tại có nhiều phương pháp

phát hiện cộng đồng, cũng như vô số biến thể và cải tiến. Chúng tôi đề cập đến một số công trình thu thập và so sánh một số lượng lớn các phương pháp.

Nhiều cuộc khảo sát nhóm các phương pháp theo nền tảng lý thuyết/khái niệm của chúng. Ví dụ, Rosvall et al. [6] phương pháp phát hiện cộng đồng nhóm dưới bốn góc độ:

- quan điểm dựa trên mặt cắt, nhằm mục đích giảm thiểu số lượng cạnh giữa các đỉnh;
- quan điểm phân cụm, trong đó tìm thấy các nhóm nút chặt chẽ, dày đặc;
- quan điểm tương đương ngẫu nhiên, trong đó suy ra các nhóm bằng cách sử dụng các mô hình thống kê (như mô hình khối ngẫu nhiên);
- quan điểm dựa vào động lực học, dựa trên cách cấu trúc mô-đun tác động đến sự phát triển của các quy trình trên mạng.

Có nhiều cách phân loại khác, xem [6, 7, 8]. Mặt khác, một số nghiên cứu so sánh các thuật toán phát hiện cộng đồng bằng cách chạy chúng trên một số lượng lớn mạng và phân tích kết quả. Dao et al. [9] phân loại các phương pháp thành năm nhóm chính: dựa trên loại bỏ cạnh, tối ưu hóa mô đun, dựa trên quy hoạch động, dựa trên suy luận thống kê và nhóm cuối cùng gồm các phương pháp khác không thuộc bốn nhóm còn lại. Các phương pháp đó được chạy trên hơn 100 mạng từ nhiều miền khác nhau, sau đó được so sánh dựa trên thời gian chạy, số lượng cộng đồng được tìm thấy và quy mô cộng đồng, chất lượng cộng đồng và sự tương đồng giữa các phân vùng được tạo. Kết quả chi tiết được đưa ra, có ý nghĩa cho việc lựa chọn một phương pháp phù hợp trong thực tế. Các nghiên cứu thực nghiệm khác, với nhiều cách tiếp cận khác nhau, bao gồm [10, 11, 12].

1.3. Nội dung chính của luận văn

Luận văn có hai chủ đề chính: thuật toán phân cụm cho đồ thị lớn và đồ thị lớn hai phần.

Trong chương 2 chúng tôi trình bày về một thuật toán phân cụm cho đồ thị lớn đó là thuật toán dựa vào động lực học khoảng cách (Distance Dynamics).

Trong chương 3 chúng tôi tiếp tục trình bày về thuật toán dựa vào động lực học khoảng cách ở trên đồ thị hai phần. Ngoài ra trong chương này chúng tôi trình bày một số thuật toán khác cho đồ thị hai phần như thuật toán: ComSim (sử dụng tính tương tự của chu trình và đỉnh).

Cuối cùng trong chương 4 chúng tôi sẽ so sánh các thuật toán này với một số các thuật toán phổ biến khác thông qua phân tích ví dụ cụ thể và thực hành chạy các thuật toán trên Python (trình bày ở phụ lục của luận văn).

Chương 2
CÁC THUẬT TOÁN PHÂN CỤM CHO ĐỒ THỊ LỚN

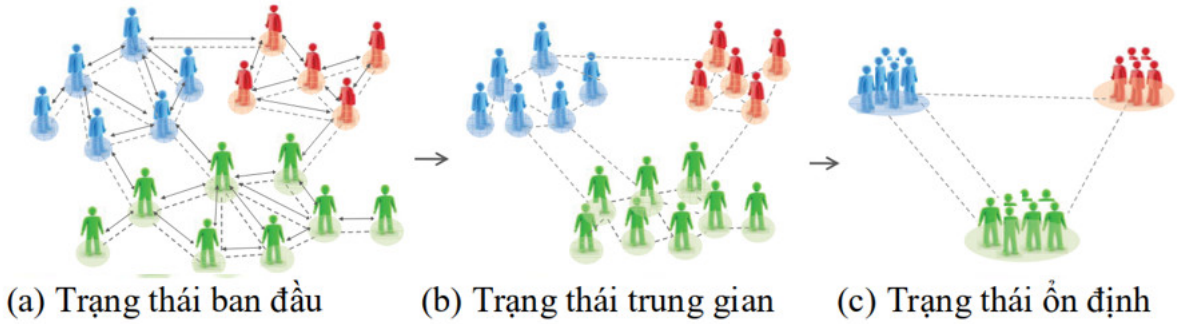
Trong chương này, dựa trên bài báo [13] của Junming Shao, Zhichao Han, Qinli Yang, TaoZhou chúng ta xây dựng khoảng cách Jaccard trên đồ thị. Từ đó, chúng ta đưa ra cách tính động lực học khoảng cách trên đồ thị và chúng ta đưa ra thuật toán Attractor để phân cụm đồ thị dựa trên khoảng cách chúng ta vừa đưa ra.

2.1. Động lực khoảng cách so với tiêu chí cộng đồng do người dùng xác định

Hiện nay, nhiều tiêu chí đã được đề xuất để đánh giá cấu trúc cộng đồng theo các quan điểm khác nhau và mỗi tiêu chí đều có những ưu và nhược điểm riêng. Trong nghiên cứu này, thay vì giới thiệu tiêu chí mới do người dùng xác định, chúng tôi trình bày một phương pháp phát hiện cộng đồng mới dựa trên động lực khoảng cách. Tiêu chí cơ bản là hình dung mạng như một hệ thống động và tìm hiểu khoảng cách động giữa các đỉnh liền kề để khám phá cấu trúc cộng đồng của nó. So với hầu hết các thuật toán hiện có, ngoại trừ cách thức sinh động để khám phá cộng đồng, quan điểm mới còn có thêm một số điểm khởi sắc đáng mong đợi. (a) Khám phá động lực học khoảng cách cung cấp một hình ảnh trực quan và toàn diện để mô hình hóa động lực học mạng trong thế giới thực. Ví dụ, một cộng đồng (ví dụ: mạng lưới tình bạn) thường được thiết lập và tăng cường dựa trên các mối quan hệ thông qua tương tác (ví dụ: các hoạt động xã hội trong mạng lưới tình bạn). (b) Không sử dụng các biện pháp do người dùng xác định, các cộng đồng được phát hiện tự động do cấu trúc liên kết địa phương bên trong của mạng. (c) Cái nhìn sâu sắc về động lực học khoảng cách cũng cung cấp một cách tổng quát để khai thác mạng trong không gian số liệu thay vì không gian vectơ. Điều này khá có lợi cho việc phân tích mạng vì thông tin của các mạng trong thế giới thực mà chúng ta thường có được là các mẫu kết nối của chúng.

Sau đây, chúng tôi bắt đầu với một số định nghĩa cần dùng đến, sau đó một mô hình tương tác được đề xuất trong mục 2.1.3, mục 2.1.4 trình bày chi tiết thuật toán Attractor

và chúng tôi phân tích độ phức tạp về thời gian của nó trong mục 2.1.5.



Hình 1: Minh họa phát hiện cộng đồng dựa trên động lực học khoảng cách. (a) Một mạng xã hội, trong đó các đường đứt nét biểu thị mối quan hệ giữa những người với nhau và các mũi tên biểu thị các tương tác trực tiếp lẫn nhau dựa trên các mối quan hệ của họ. (b) Dựa vào mô hình tương tác được đề xuất, "khoảng cách" giữa mọi người sẽ thay đổi theo thời gian, trong đó những người trong cùng một cộng đồng có xu hướng dần dần di chuyển đến gần nhau trong khi những người trong các cộng đồng khác nhau sẽ cách xa nhau. (c) Trạng thái ổn định của con người xét về "khoảng cách": ba cộng đồng trực giác.

2.2. Kiến thức liên quan

Với mục đích phát hiện cộng đồng, một số định nghĩa cần thiết trước tiên được giới thiệu.

Định nghĩa 2.1. Giả sử $G = (V, E, W)$ là một đồ thị có hướng, trong đó V là tập hợp đỉnh, E là tập hợp cạnh; W là tập trọng số tương ứng với cạnh. $e = \{u, v\} \in E$ chỉ ra một kết nối giữa các đỉnh u và v. $w(u, v)$ biểu diễn cho trọng số của cạnh e. $\forall e = \{u, v\} \in E, w(u, v) = 1$, trong trường hợp đồ thị không có trọng số.

Định nghĩa 2.2. (Lân cận của đỉnh u) Cho một đồ thị vô hướng $G = (V, E, W)$, lân cận của đỉnh $u \in V$ là tập hợp $N(u)$ chứa đỉnh u và các đỉnh lân cận của nó.

$$N(u) = \{u \in V\{u, v\} \in E\} \cup \{u\} \tag{2.1}$$

Dựa trên hai định nghĩa, chúng tôi tiếp tục sử dụng khoảng cách Jaccard [14] để xác định khoảng cách ban đầu giữa hai đỉnh liền kề. Lựa chọn biện pháp này chủ yếu có hai lý do. Đầu tiên, khoảng cách Jaccard cho chúng ta một cách trực quan để mô tả tính tương tự của đỉnh. Nói chung, hai đỉnh càng có nhiều đỉnh lân cận chung thì chúng càng có nhiều điểm chung và liên kết chặt chẽ hơn. Thứ hai, khoảng cách Jaccard được tính theo kiểu địa phương và do đó hiệu quả về mặt thời gian.

Định nghĩa 2.3. (Khoảng cách Jaccard) Cho một đồ thị vô hướng $G = (V, E, W)$, khoảng cách Jaccard của hai đỉnh u và v được xác định là:

$$d(u, v) = 1 - \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{2.2}$$

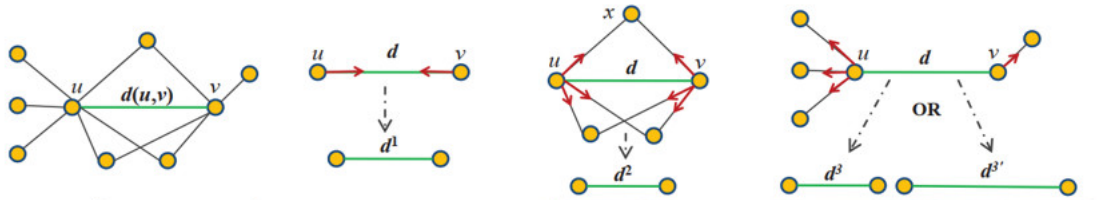
Đối với đồ thị có trọng số, khoảng cách Jaccard của hai đỉnh u và v được mở rộng thêm như sau:

$$d(u, v) = 1 - \frac{\sum_{x \in N(u) \cap N(v)} (w(u, x) + w(v, x))}{\sum_{\{x,y\} \in E; x,y \in N(u) \cup N(v)} w(x, y)} \tag{2.3}$$

2.3. Mô hình tương tác địa phương

Để khám phá cấu trúc cộng đồng trong các mạng dựa trên động lực khoảng cách, chúng ta nên xây dựng một mô hình tương tác phù hợp. Do đó, phạm vi tương tác và mô hình tương tác cần được xem xét đầu tiên.

Phạm vi tương tác. Để xác định cấu trúc cộng đồng trong các mạng, việc khám phá cấu trúc liên kết địa phương là điều cần thiết. Do đó, thay vì quan sát tất cả các tương tác, chúng tôi tập trung vào động lực khoảng cách theo cách địa phương. Rõ ràng, các kết nối bên trong (các cạnh) của các mạng trong thế giới thực mang lại một cách tự nhiên để mô hình hóa phạm vi tương tác. Chính xác là đối với mỗi đỉnh, nó tương tác một cách tự nhiên với các đỉnh liền kề của nó.



(a) Đồ thị (b) Ảnh hưởng từ các đỉnh liên kết trực tiếp (c) Ảnh hưởng từ các đỉnh lân cận chung (d) Ảnh hưởng từ các đỉnh lân cận riêng

Hình 2: Minh họa sự thay đổi khoảng cách giữa các đỉnh bị ảnh hưởng bởi ba kiểu tương tác riêng biệt.

Các mẫu tương tác. Sau khi chỉ định phạm vi tương tác, bước quan trọng tiếp theo là xác định các kiểu tương tác giữa các đỉnh để mô phỏng động lực khoảng cách. Chính thức, đặt $e = \{u, v\} \in E$ là một cạnh giữa hai đỉnh liền kề u và v, và $d(u, v)$ là khoảng cách ban đầu của nó. Rõ ràng, bất kỳ thay đổi nào của khoảng cách $d(u, v)$ đều là kết quả của sự thay đổi của đỉnh u và đỉnh v. Trên thực tế, có ba tình huống riêng biệt cho phép ảnh hưởng đến khoảng cách $d(u, v)$, dựa vào cấu trúc tô pô địa phương của nó (xem Hình 2). Trong phần

sau đây, chúng tôi sẽ trình bày chi tiết cách khoảng cách thay đổi tương ứng trong ba tình huống khác nhau.

Trường hợp 1: Ảnh hưởng từ các đỉnh liên kết trực tiếp. Khoảng cách $d(u, v)$ giữa đỉnh u và đỉnh v rõ ràng bị ảnh hưởng bởi hai đỉnh liên kết trực tiếp u và v. Thông qua sự tương tác với nhau, đỉnh này hút đỉnh kia di chuyển về phía mình và do đó làm giảm $d(u, v)$ (xem Hình 2 (b)). Giống như một mạng lưới tình bạn, mỗi người đều ảnh hưởng đến những người quen biết của họ và có xu hướng tăng dần tính gắn kết (tức là "khoảng cách" sẽ giảm đi). Chính thức, để biểu diễn cho sự thay đổi của khoảng cách $d(u, v)$, ta định nghĩa DI, biểu thị ảnh hưởng từ tương tác của các đỉnh liên kết trực tiếp, như sau:

$$DI = -\left(\frac{f(1 - d(u, v))}{deg(u)} + \frac{f(1 - d(u, v))}{deg(v)}\right) \tag{2.4}$$

trong đó $\deg(u)$ (số cạnh liên thuộc với u) là bậc của đỉnh u, $f(.)$ là một hàm ghép và sin() được sử dụng trong nghiên cứu này. $1 - d(u, v)$ biểu thị sự giống nhau giữa u và v và hai đỉnh càng giống nhau thì chúng sẽ có ảnh hưởng lẫn nhau càng cao. Thuật ngữ: $1/\deg(u)$ được gọi là hệ số chuẩn hóa, được dùng để xem xét các ảnh hưởng khác nhau giữa các đỉnh được liên kết với các mức độ khác nhau. Cụ thể, các đỉnh có nhiều liên kết hơn sẽ khó bị ảnh hưởng hơn so với các đỉnh có ít liên kết hơn. Lấy mạng người hướng dẫn làm ví dụ. Một người giám sát thường liên kết với nhiều sinh viên trong khi một sinh viên chỉ kết nối với người giám sát của mình. Trong tình huống này, người giám sát có thể có ảnh hưởng lớn đến từng học sinh trong khi ảnh hưởng đối với người giám sát từ mỗi học sinh là tương đối thấp. Tóm lại, hai đỉnh có liên kết trực tiếp sẽ có xu hướng lại gần nhau hơn.

Trường hợp 2: Ảnh hưởng từ đỉnh lân cận chung. Chúng tôi xem xét khả năng thứ hai: ảnh hưởng từ những đỉnh lân cận chung $CN = (N(u) - u) \cap (N(v) - v)$ của các đỉnh u và v (Hình 2(c)). Vì các đỉnh lân cận chung có cả hai liên kết với hai đỉnh u và v nên chúng hút hai đỉnh này và do đó dẫn đến sự thay đổi khoảng cách $d(u, v)$. Cụ thể, mỗi đỉnh lân cận chung sẽ thu hút cả đỉnh u và đỉnh v để di chuyển về phía chính nó và do đó dẫn đến việc giảm khoảng cách $d(u, v)$ (xem Hình 2(c)). Chính thức, chúng tôi xác định sự thay đổi của $d(u, v)$ từ ảnh hưởng của các đỉnh lân cận chung, CI, như sau:

$$CI = -\Sigma_{x \in CN}\left(\frac{1}{deg(u)}.f(1 - d(x, u).(1 - d(x, v))) + \frac{1}{deg(v)}.f(1 - d(x, v).(1 - d(x, u))\right) \tag{2.5}$$

Ở đây, hai thuật ngữ $(1 - d(x, v))$ và $(1 - d(x, u))$ cho mỗi đỉnh lân cận chung được sử dụng để định lượng thêm mức độ ảnh hưởng so với ảnh hưởng từ các đỉnh được liên kết trực tiếp.

Tóm lại, hai đỉnh có càng nhiều đỉnh lân cận chung càng có xu hướng lại gần nhau hơn. Ví dụ: khi xem xét một lân cận chung x tương tác với đỉnh u (xem Hình 2(c)), nếu x và v càng giống nhau thì ảnh hưởng từ x lên u càng lớn tương tự như ảnh hưởng từ v. Về mặt lý thuyết, một khi độ giống nhau giữa x và v bằng một (nghĩa là chúng có thể được xem như cùng một đỉnh), ảnh hưởng của đỉnh x trên khoảng cách $d(u, v)$ chỉ đơn giản là chuyển vào mẫu đầu tiên.

Trường hợp 3: Ảnh hưởng từ các đỉnh lân cận riêng: Mẫu tương tác thứ ba xảy ra khi tồn tại một số lân cận chỉ thuộc về đỉnh u hoặc v, lần lượt là $EN(u) = \Sigma(u) - \Sigma(u) \cup \Sigma(v), EN(v) = \Sigma(v) - \Sigma(u) \cup \Sigma(v)$. Mặc dù, giống như mẫu 1 và mẫu 2, mỗi đỉnh lân cận riêng của u thu hút u di chuyển đến gần chính nó, không biết liệu đỉnh u bị thu hút để di chuyển đến gần đỉnh v hay bị thu hút để di chuyển ra xa v (xem Hình 2 (d)). Để xác định ảnh hưởng tích cực hoặc tiêu cực của các lân cận riêng đối với khoảng cách, một cách tìm hiểu chúng dựa trên sự tương đồng được đề xuất. Vấn đề cơ bản là điều tra xem mỗi đỉnh lân cận riêng của đỉnh u có giống với đỉnh v hay không và ngược lại. Nếu đỉnh lân cận riêng của đỉnh u tương tự với đỉnh v, thì sự di chuyển của đỉnh u về phía đỉnh lân cận riêng sẽ làm giảm khoảng cách $d(u, v)$. Tương tự, nếu đỉnh lân cận riêng của u không tương tự với đỉnh v, thì sự di chuyển của đỉnh u về phía đỉnh lân cận riêng sẽ dẫn đến hiệu ứng ngược lại: di chuyển ra xa khỏi đỉnh v. Do đó, ở đây chúng tôi giới thiệu tham số lực dính λ, để xác định ảnh hưởng cơ bản như sau. Tham số lực dính λ sẽ được thảo luận thêm trong Phần 2.3

$$p(x,u) = \begin{cases} (1 - d(x,v)) & (1 - d(x,v)) \geq \lambda \\ (1 - d(x,v)) - \lambda & \text{trường hợp khác} \end{cases} \tag{2.6}$$

trong đó $\rho(x, u)$ đặc trưng cho mức độ ảnh hưởng tích cực hoặc tiêu cực đến khoảng cách $d(u, v)$. Khi đó, sự thay đổi của $d(u, v)$ ảnh hưởng bởi các đỉnh lân cận riêng, EI, được định nghĩa như sau: đặc trưng cho mức độ ảnh hưởng tích cực hoặc tiêu cực đến khoảng cách $d(u, v)$.

$$EI = -\Sigma_{x \in EN(u)}(\frac{1}{deg(u)}.f(1 - d(x,u)).p(x,u)) \tag{2.7}$$
$$-\Sigma_{y \in EN(v)}(\frac{1}{deg(v)}.f(1 - d(y,v)).p(y,v))$$

Cuối cùng, bằng cách xem xét ba kiểu tương tác cùng nhau, động lực học của khoảng cách $d(u, v)$ giữa các đỉnh u và v theo thời gian bị chi phối bởi:

$$d(u, v, t + 1) = d(u, v, t) + DI(t) + CI(t) + EI(t) \tag{2.8}$$

ở đây $d(u, v, t+1)$ là khoảng cách được gia hạn tại bước thời gian $t+1$. $DI(t), CI(t)$ và $EI(t)$ lần lượt mô tả sự thay đổi khoảng cách từ các đỉnh được liên kết trực tiếp, các đỉnh lân cận chung và các đỉnh lân cận riêng.

2.4. Thuật toán Attractor

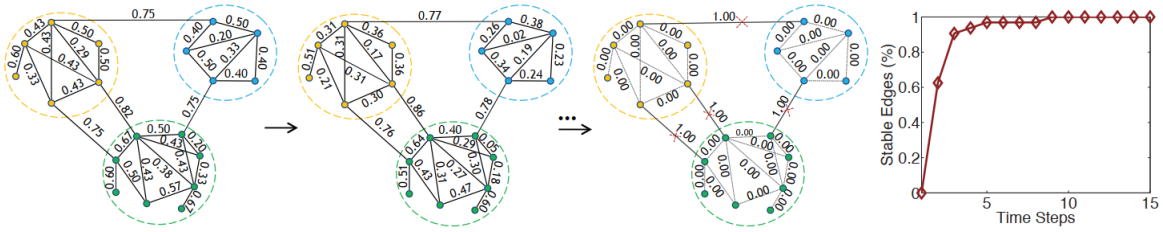
Trong phần này, chúng tôi trình bày chi tiết thuật toán Attractor. Tương tác động.

Dựa trên mô hình tương tác (xem phương trình (2.8)), động lực học khoảng cách có thể được mô phỏng, chủ yếu bao gồm các bước sau:

a) Tại thời điểm ban đầu ($t = 0$), không có bất kỳ tương tác nào, mỗi cạnh được liên kết với một khoảng cách ban đầu. Ở đây, giá trị ban đầu được tính theo khoảng cách Jaccard với Định nghĩa 2.2 hoặc Định nghĩa 2.3.

b) Khi thời gian thay đổi, dựa trên cấu trúc tô pô địa phương, động lực học của từng khoảng cách được mô phỏng theo ba mẫu tương tác được đề xuất (Phương trình (2.4), Phương trình (2.5) và Phương trình (2.7)). Nhờ các ảnh hưởng định hướng tô pô, khoảng cách giữa các đỉnh chia sẻ cùng một cộng đồng có xu hướng giảm dần trong khi khoảng cách giữa các cộng đồng khác nhau tăng dần.

c) Cuối cùng, tất cả các khoảng cách sẽ hội tụ và có thể dễ dàng thu được các cộng đồng bằng cách loại bỏ các cạnh có khoảng cách tối đa (tức là $d(u, v) = 1$).



Hình 3: Minh họa về động lực học khoảng cách. (a) Biểu diễn đồ thị của mạng xã hội trong Hình 1(a), trong đó các số trên các cạnh biểu thị khoảng cách ban đầu giữa các đỉnh được kết nối. (b) Khoảng cách giữa các đỉnh được cập nhật sau một bước thời gian. (c) Trạng thái cuối cùng của mạng.

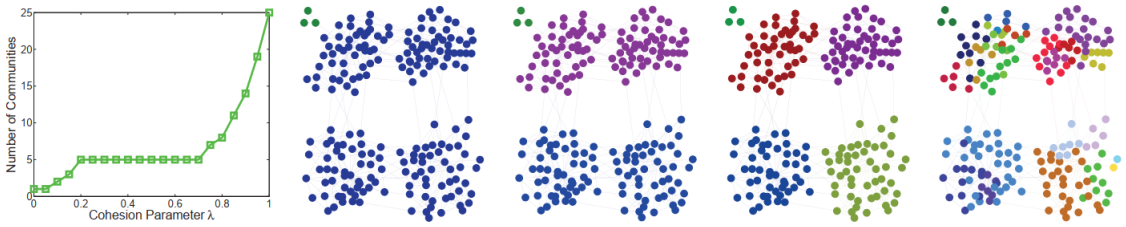
Để minh họa, Hình 3(a)-(c) hiển thị ba trạng thái cho mạng xã hội của Hình 1 từ $t = 0$ đến $t = 9$ trong quá trình tương tác động địa phương. $T = 0$ biểu thị khoảng cách ban đầu giữa các đỉnh được kết nối (Hình 3(a)). Kể từ thời điểm đó, mỗi đỉnh tương tác với các đỉnh lân cận và ảnh hưởng đến khoảng cách tương ứng dựa trên mô hình tương tác được đề xuất

(xem Phương trình 2.8) và khoảng cách mới sau một bước thời gian được minh họa rõ hơn trong Hình 3 (b). Sau chín bước, tất cả các khoảng cách hội tụ, 0 hoặc 1 và ba cộng đồng được xác định một cách tự nhiên bằng cách cắt bỏ tất cả các cạnh có khoảng cách bằng 1.

Phát hiện các cộng đồng nhỏ hoặc sự bất thường. Trong các mạng trong thế giới thực, thường tồn tại nhiều cộng đồng với nhiều quy mô khác nhau. Đặc biệt là trong các mạng quy mô lớn, quy mô của một phần lớn cộng đồng thường nhỏ [15]. Tuy nhiên, đối với nhiều thuật toán phát hiện cộng đồng truyền thống, chẳng hạn như Modularity hoặc Ncut, chúng có xu hướng phân vùng toàn bộ mạng thành các nhóm có kích thước tương đối bằng nhau với kích thước cụm không nhỏ hơn $\sqrt{n}$ (n là số đỉnh trong mạng) [15] và không tìm thấy các cộng đồng nhỏ do vấn đề gọi là "giới hạn độ phân giải" [16]. Đối với thuật toán Attractor, vì nó mô phỏng động lực học khoảng cách và không dựa vào bất kỳ tiêu chí nào do người dùng xác định, nên nó cho phép tìm kiếm các cộng đồng với kích thước tùy ý trong mạng một cách trực quan. Do đó, nó cũng cung cấp một cách đầy hứa hẹn để xử lý các điểm bất thường/ngoại lệ. Trong trường hợp này, các điểm bất thường được hiểu là các đỉnh liên kết với nhiều đỉnh khác hoặc các đỉnh bất thường bị cô lập khỏi tất cả các đỉnh khác theo thời gian và cuối cùng sẽ tự động bật ra.

Algorithm 1 Thuật toán Attractor

Đầu vào: $G = (V, E, W), \lambda$
Khởi tạo khoảng cách
for mỗi cạnh $e = \{u, v\} \in E$ **do**
    tính toán khoảng cách ban đầu d_e^0 sử dụng phương trình 2.3
    for mỗi đỉnh $x \in EN(u)$ **do do**
        tính khoảng cách d_{vy}^0 sử dụng phương trình 2.3;
    end for
    for mỗi đỉnh $y \in EN(v)$ **do**
        tính khoảng cách d_{vy}^0
    end for
end for
Động lực học khoảng cách
Flag=FALSE
while Flag **do**
    Flag=FALSE
    for mỗi cạnh $e = \{u, v\} \in E$ **do**
        if $0 < d_e^t < 1$ **then**
            Tính DI_e^t, CI_e^t, EI_e^t sử dụng phương trình 2.4, 2.5 , 2.7;
            $\Delta d_e^t = DI_e^t + CI_e^t + EI_e^t$;
            if $\Delta d_e^t \neq 0$ **then**
                tính toán khoảng cách thay đổi theo thời gian
                $\Delta d_e^{t+1} = d_e^t + \Delta d_e^t$
                if $d_e^{t+1} > 1$ **then**
                    $d_e^{t+1} = 1$;
                end if
                if $d_e^{t+1} < 0$ **then**
                    $d_e^{t+1} = 0$;
                end if
                Flag = TRUE
            end if
        end if
    end for
end while
Tìm cộng đồng
for mỗi cạnh $e = \{u, v\} \in E$ **do**
    if $d_e^{t+1} = 1$ **then**
        xóa cạnh e khỏi mạng;
    end if
end for
tìm các thành phần kết quả (cộng đồng) C;
Đầu ra: C;



(a) λ vs #C (b) $\lambda = 0.1$ (c) $\lambda = 0.15$ (d) $\lambda \in [0.2 - 0.7]$ (e) $\lambda = 1$

Hình 4: Độ nhạy của tham số liên kết λ khi phát hiện cộng đồng.

Tham số liên kết λ. Đối với thuật toán Attractor, tham số liên kết λ được sử dụng để xác định ảnh hưởng tương tác tích cực hoặc tiêu cực đối với khoảng cách từ các lân cận loại trừ (xem phương trình (2.6)). Nói chung, với giá trị cao hơn của λ, nó mang lại nhiều cộng đồng hơn trong khi tạo ra các cộng đồng lớn hơn với giá trị thấp hơn của λ. Bằng cách điều chỉnh tham số gắn kết λ, Attractor cho phép phân tích cấu trúc cộng đồng từ thô đến tinh. Hơn nữa, λ cung cấp nhiều thông tin và dễ điều chỉnh so với các thuật toán yêu cầu chỉ định số lượng cụm. Hình 4(a) biểu thị số lượng cộng đồng tìm được với các λ khác nhau nằm trong khoảng từ 0 đến 1 trên mạng tổng hợp. Từ biểu đồ này, chúng ta có thể thấy rằng Attractor cho phép tạo ra cách phân cụm tốt với tham số λ trên phạm vi ổn định dài $(0,2 - 0,7)$. Các kết quả phân cụm đối với các tham số riêng biệt được minh họa rõ hơn trong Hình 4 (b) đến Hình 4 (e). Các thử nghiệm mở rộng chứng minh thêm rằng Attractor không nhạy cảm với kết quả phân cụm và thường tạo ra kết quả tốt trong phạm vi $\lambda = [0,4; 0,6]$. Cuối cùng, giả code của Attractor được đưa ra trong Thuật toán 1.

2.5. Phân tích độ phức tạp

Để tính độ phức tạp của động lực khoảng cách, khoảng cách ban đầu của hai đỉnh được liên kết bất kỳ trong mạng là bắt buộc và do đó thời gian tính toán là $O(|E|)$. Ngoài ra, đối với tương tác động địa phương, Attractor cũng cần tính toán khoảng cách jaccard tương ứng cho các lân cận riêng (Thuật toán 1 (Dòng 5-10)). Độ phức tạp về thời gian là $O(k.|E|)$, trong đó k xấp xỉ là số lân cận riêng trung bình của hai đỉnh được liên kết. Trong quá trình tương tác động địa phương, vì tất cả các khoảng cách đã tồn tại, Attractor chỉ cần gọi lại các khoảng cách này ở thời gian trước đó mà không cần tính toán khoảng cách và do đó độ phức tạp thời gian là $O(T.|E|)$. Tổng cộng, độ phức tạp tính toán về thời gian là $O(|E| + k.|E| + T|E|)$, trong đó T là số bước thời gian. Trong hầu hết các trường hợp, T nhỏ với $3 \leq T \leq 50$.

Chương 3
CÁC THUẬT TOÁN PHÂN CỤM CHO ĐỒ THỊ LỚN HAI PHẦN

Trong chương này, chúng tôi phân tích và hiểu thấu đáo hai bài báo: [17] của Hong-liang Sun, Eugene Ch'ng, Xi Yong, Jonathan M. Garibaldi, Simon See và Duan-bing Chen; [18] của Raphael Tackx, Fabien Tarissan và Jean-Loup Guillaume. Chúng tôi trình bày khoảng cách Jaccard trên đồ thị hai phần dựa trên khoảng cách Jaccard trình bày trong chương 3 và thuật toán BiAttactor được cải thiện và phát triển từ thuật toán Attactor được trình bày trong 3.1. Ngoài ra chũng tôi trình bày thuật toán COMSIM sử dụng chu trình và sự về sự tương tự của các đỉnh và thuật toán này được áp dụng trong bài toán phân cụm đồ thị hai phần trình bày trong phần 3.2.

3.1. Phát hiện cộng đồng trong mạng hai phần bằng động lực học khoảng cách

3.1.1. Động lực khoảng cách trong Unipartite Networks

Trong phần này, chúng tôi mô tả ngắn gọn thuật toán Attractor để khám phá các cộng đồng của các mạng đơn phương sử dụng động lực học khoảng cách. Attractor là nền tảng của BiAttractor được đề xuất của chúng tôi trong phần tiếp theo. Ý tưởng về Attractor được lấy cảm hứng từ quan điểm hình thành cộng đồng từ xã hội học. Một cộng đồng bạn bè thường được thiết lập và tăng cường do các tương tác nhất quán. Những tương tác như vậy thường được thúc đẩy bởi lợi ích của ý chí, chia sẻ lợi ích chung,... Tương tác giữa những người trong xã hội đưa ra một hình ảnh sống động về khoảng cách động giữa các đỉnh trong khoa học mạng. Do đó, Attractor đã đề xuất để khám phá các cộng đồng từ các mạng bên dưới bằng cách sử dụng động lực học khoảng cách [13]. Mạng được xem như một hệ thống động trong đó các động lực cạnh đã được nghiên cứu. Do sự tương tác của các đỉnh với các đỉnh lân cận của chúng, khoảng cách giữa chúng tăng dần theo thời gian và cuối cùng đạt đến sự hội tụ. Các cộng đồng có thể được phát hiện một cách tự nhiên bằng cách loại bỏ các cạnh liên cộng đồng.

Tiếp theo, các bước chính của Attractor đã được giới thiệu. Cho một mạng vô hướng và không trọng số $G = (V; E)$ trong đó V là tập các đỉnh và E là tập các cạnh. Attractor nhằm mục đích tìm một phân vùng của V thành các cộng đồng $V_1; V_2; ...; V_C$ dựa trên động lực học khoảng cách.

a) Ban đầu, mỗi cạnh được liên kết với một khoảng cách ban đầu theo hệ số Jaccard.

b) Theo thời gian, khoảng cách của mỗi cạnh thay đổi do tác động qua lại giữa các đỉnh lân cận địa phương. Các mẫu tương tác bao gồm ảnh hưởng từ các đỉnh được kết nối trực tiếp, ảnh hưởng từ các đỉnh lân cận chung và các đỉnh lân cận riêng.

c) Cuối cùng, mọi khoảng cách đều đạt trạng thái hội tụ. Các cạnh được liên kết với số 0 biểu thị các cạnh trong cộng đồng. Những cái được liên kết với một cạnh chỉ ra giữa các cộng đồng đã bị xóa để hình thành các cộng đồng một cách tự nhiên.

Attractor có tham số λ để xác định kích thước và số lượng cộng đồng. Nó ảnh hưởng đến những đỉnh lân cận riêng như một ngưỡng tác động tích cực hoặc tác động tiêu cực. Độ phức tạp thời gian tuyến tính $O(|E|)$ cho phép nó được áp dụng cho các mạng thưa thớt có kích thước rất lớn trong một thời gian ngắn.

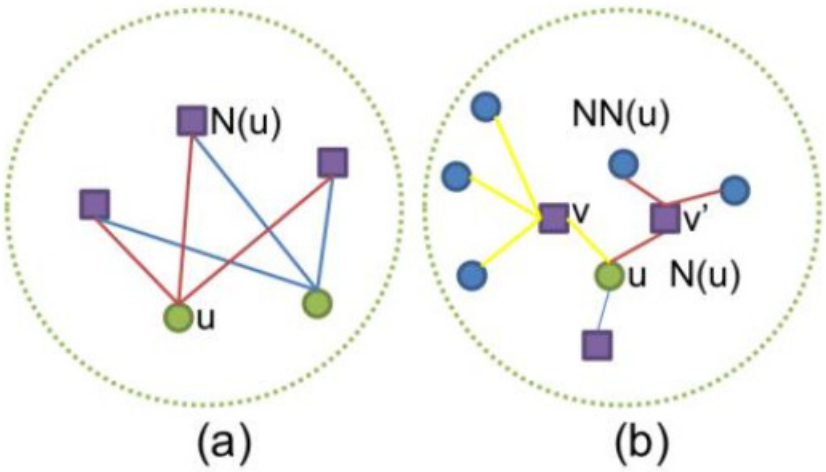
Tuy nhiên, Attractor chỉ được áp dụng cho các mạng đơn phương chứa một loại đỉnh. Vì vậy, chúng tôi khái quát hóa nó để xử lý các mạng hai phần trong phần tiếp theo.

3.1.2. Động lực khoảng cách trong mạng hai phần

Trong phần này, chúng tôi mở rộng Attractor thành BiAttractor để giải quyết việc phát hiện cộng đồng trong các mạng hai phần. Một số định nghĩa liên quan sẽ được giải thích như sau.

$G = (U, V, E)$ là một đồ thị hai phía vô hướng và không trọng số, trong đó U và V đại diện cho hai tập hợp các loại đỉnh khác nhau. Cạnh e thuộc E kết nối các loại đỉnh khác nhau. Không có cạnh giữa các đỉnh từ cùng một tập hợp U hoặc V. Attractor trong phần trước phụ thuộc vào lân cận của đỉnh và Khoảng cách Jaccard. Tương tự, phương pháp BiAttractor của chúng tôi phụ thuộc vào các lân cận bậc hai của đỉnh và khoảng cách Jaccard địa phương (LJD) mới.

Tiếp theo, chúng sẽ được giới thiệu chi tiết.



Hình 5: Các đỉnh lân cận và các đỉnh lân cận bậc hai của các đỉnh đã cho. Hình tròn và hình vuông biểu thị hai loại đỉnh trong mạng hai phần. (a) Đỉnh u ***của hình tròn màu lục và các lân cận của nó*** $N(u)$ ***được biểu thị bằng các hình vuông màu tím. (b) Các vòng tròn màu xanh biểu thị đỉnh lân cận bậc hai*** $NN(u)$ ***của*** u***.***

Định nghĩa 3.1. Vùng lân cận của đỉnh u bao gồm các đỉnh lân cận của u. Tương tự, chúng ta có thể có được lân cận của đỉnh v. Trong các mạng hai phần, các lân cận được kết nối trực tiếp của đỉnh u thuộc về tập V được hiển thị trong hình 5 (a). Nó tương tự như các đỉnh lân cận được kết nối trực tiếp của đỉnh v.

$$N(u) = \{v | v \in V, (u, v) \in E\} \tag{3.1}$$

$$N(v) = \{u | u \in U, (u, v) \in E\} \tag{3.2}$$

Định nghĩa 3.2. (Lân cận bậc hai của đỉnh u và đỉnh v)

Phần mở rộng từ đỉnh lân cận đến đỉnh lân cận bậc hai được thể hiện trong Hình 5 (b). Nếu chúng ta định nghĩa $N(u)$, các lân cận của đỉnh u, là các đỉnh liền kề của nó.

Khi đó các đỉnh lân cận bậc hai $NN(u)$ được định nghĩa là các đỉnh lân cận của y, trong đó y là các đỉnh lân cận của u. Định nghĩa của $NN(v)$ tương tự như $NN(u)$.

$$NN(u) = \{x | x \in N(y), y \in N(u), x \in U, y \in V \text{ và } u \in U\} \tag{3.3}$$

$$NN(v) = \{x | x \in N(y), y \in N(v), x \in V, y \in U \text{ và } v \in V\} \tag{3.4}$$

Định nghĩa 3.3. (Khoảng cách Jaccard)

Cho một đồ thị hai phần vô hướng và không có trọng số $G = (U, V, E)$, khoảng cách Jaccard của đỉnh u và đỉnh v được định nghĩa như bên dưới trong biểu thức 3.5. $N(u) \cap N(v)$ là tập rỗng nếu u liên thông với v vì $N(u)$ và $N(v)$ là các loại đỉnh khác nhau.

$$d_{jac}(u, v) = 1 - \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{3.5}$$

Định nghĩa 3.4. (Khoảng cách Jaccard địa phương)

Khoảng cách Jaccard hoạt động tốt trong Attractor vì các đỉnh lân cận chung $N(u) \cap N(v)$ tồn tại trong các mạng đơn phương. Tuy nhiên, nó không giống như vậy trong các mạng hai phần. $N(u)$ và $N(v)$ là các loại đỉnh khác nhau và $(N(u) \cap N(v)) = \emptyset$. Tuy nhiên, u đỉnh lân cận bậc hai $NN(u)$ là cùng loại với $N(v)$ đỉnh lân cận của v. Do quan điểm này, chúng tôi đề xuất khoảng cách Jaccard địa phương để xử lý việc phát hiện cộng đồng trong các mạng hai phần.

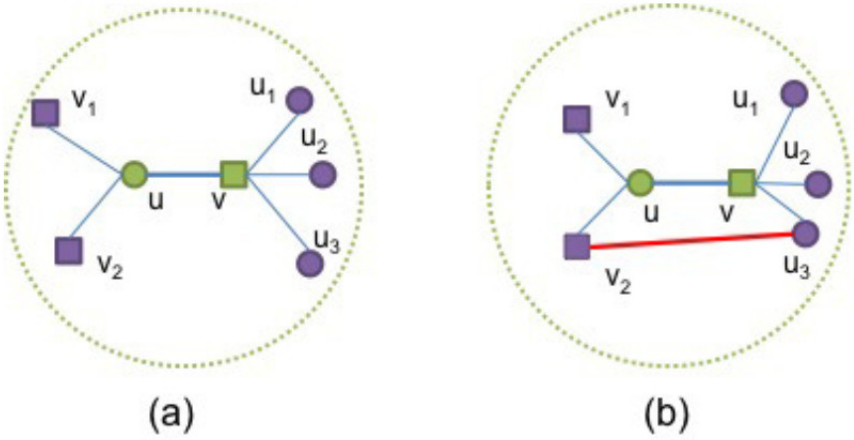
Khoảng cách Jaccard địa phương (LJD) giữa đỉnh u và v được hiển thị trong biểu thức 3.6 Các đỉnh lân cận riêng của u được định nghĩa là $EN(u) = N(u) - \{v\}$. Tương tự, chúng ta có thể lấy các đỉnh lân cận riêng của v là $EN(v) = N(v) - \{u\}$. Khoảng cách Jaccard giữa đỉnh u và các đỉnh lân cận riêng của v được tính tổng và chia cho số lân cận riêng của nó $|EN(v)|$. Vì các đỉnh lân cận của u và các đỉnh lân cận bậc hai của v là cùng một loại đỉnh. Tương tự, khoảng cách Jaccard từ đỉnh v và các đỉnh lân cận riêng của u được tính tổng và chia cho số đỉnh lân cận riêng của nó $|EN(u)|$. Giá trị trung bình của hai phần đó được định nghĩa là LJD. Ngoài ra, các đỉnh lân cận bậc hai có hiệu quả để xử lý dự đoán liên kết của các mạng phức [19, 20].

$$d(u, v) = \frac{1}{2}\left(\frac{1}{|EN(v)|}\sum_{v' \in EN(v)} d_{jac}(u, v') + \frac{1}{|EN(u)|}\sum_{u' \in EN(u)} d_{jac}(v, u')\right) \tag{3.6}$$

Để so sánh Khoảng cách Jaccard và LJD trong các mạng hai phần, một ví dụ mẫu được đưa ra dưới đây.

Như được hiển thị trong Hình 6 (a), $d_{jac}(u, v) = 1.000$ theo biểu thức 3.5. Tương tự, nó được hiển thị trong Hình 6 (b) trong đó $d_{jac}(u, v) = 1.000$. Đỉnh v_2 và u_3 có thông nhau hay không không ảnh hưởng đến d_{jac} trong G_1 và d'_{jac} trong G_2. Nhưng trên thực tế, cạnh màu đỏ giữa đỉnh v_2 và u_3 dẫn đến khoảng cách ngắn hơn giữa u và v trong G_2. Hơn nữa, chúng tôi áp dụng LJD cho cùng một mẫu. Như được hiển thị trong Hình 6 (a), $d(u, v) = 0,708$ theo biểu thức 3.6. Tương tự, nó được hiển thị trong Hình 6 (b) trong đó $d(u, v) = 0,590$.

$d(u, v)$ của G_2 nhỏ hơn $d(u, v)$ của G_1 vì cạnh đỏ giữa đỉnh v_2 và u_3 làm cho u và v gần nhau hơn.



Hình 6

Một ví dụ để so sánh Khoảng cách Jaccard và Khoảng cách Jaccard địa phương trong các mạng hai phần. (a) Một mạng hai phần mẫu $G_1 = (U_1, V_1, E_1)$, trong đó $|U_1| = 4, |V_1| = 3$ và $|E_1| = 6$. Đỉnh v_2 và u_3 đối đỉnh. (b) Một đồ thị hai phần mẫu $G_2 = (U_2, V_2, E_2)$, trong đó $|U_2| = 4, |V_2| = 3$ và $|E_2| = 7$. Đỉnh v_2 và u_3 thông nhau.

a. Mô hình tương tác

Trong phần này, các mẫu tương tác của BiAttractor sẽ được giải thích trước khi chúng tôi diễn giải thuật toán BiAttractor. BiAttractor được đưa ra bởi các mô hình tương tác xã hội trong xã hội loài người. Mọi người có xu hướng có mối quan hệ thân thiết với bạn bè của họ trong cùng một cộng đồng, nhưng giữ khoảng cách nhất định với người lạ. Mối quan hệ của họ không hoàn toàn ổn định mà thay đổi theo thời gian. Lấy cảm hứng từ ý tưởng này, chúng tôi nghĩ rằng các đỉnh gần với các đỉnh đó trong cùng một cộng đồng nhưng cách xa các đỉnh trong các cộng đồng khác nhau. Về mặt lý thuyết, các mối quan hệ chặt chẽ được biểu thị bằng các cạnh trong cộng đồng và các mối quan hệ chung được biểu thị bằng các cạnh liên cộng đồng.

Cần xác định phạm vi tương tác trước khi đưa ra các mẫu tương tác. Một đỉnh không thể tương tác với mọi đỉnh khác trong mạng bên dưới. Thay vào đó, nó tương tác với các đỉnh lân cận để tạo thành cộng đồng. Phạm vi địa phương thay vì phạm vi toàn bộ được chọn trong phương pháp của chúng tôi.

Có ba kiểu tương tác khác nhau của Attractor bao gồm ảnh hưởng của các đỉnh được

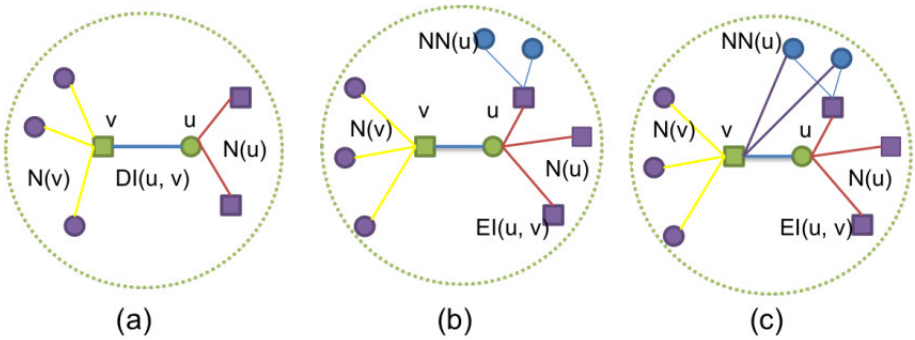
liên kết trực tiếp, ảnh hưởng của các lân cận chung và ảnh hưởng của các lân cận riêng [13]. Tuy nhiên, không có lân cận chung của u và v trong các mạng hai phần vì $N(u)$ và $N(v)$ là các loại đỉnh khác nhau. Do đó, chỉ có hai sự tương tác được nói tới trong BiAttractor bao gồm ảnh hưởng của các đỉnh được kết nối trực tiếp và ảnh hưởng của các lân cận riêng.

Định nghĩa 3.5. (Sự ảnh hưởng trực tiếp)

Khoảng cách trực tiếp giữa đỉnh u và đỉnh v được định nghĩa như bên dưới. Ảnh hưởng trực tiếp $DI(u, v)$ dẫn đến hiện tượng u và v di chuyển lại gần nhau do u và v liên thông với nhau.

$$DI(u, v) = -\left(\frac{f(1 - d(u, v))}{deg(u)} + \frac{f(1 - d(u, v))}{deg(v)}\right) \tag{3.7}$$

Nó tương tự như định nghĩa được đề xuất bởi Shao et al. [13], trong đó $f(u)$ là $\sin(u)$, $\deg(u)$ là bậc của đỉnh u. Biểu thức minh họa của khái niệm này được thể hiện trong Hình 3 (a). Chúng tôi thay đổi hàm khoảng cách $d(u, v)$ từ khoảng cách Jaccard thành Khoảng cách Jaccard địa phương vì Khoảng cách Jaccard luôn không biểu thị được độ tương tự của đỉnh trong các mạng hai phần. Lý do đã được minh họa trước đây khi chúng tôi trình bày định nghĩa về khoảng cách Jaccard địa phương.



Hình 7: Các mẫu tương tác của BiAttractor. Hình tròn và hình vuông đại diện cho các loại đỉnh khác nhau.

(a) Tương tác trực tiếp giữa đỉnh u và v được đánh dấu bằng các đường liền nét có trọng số. Hình tròn màu lục u chịu ảnh hưởng của hình vuông màu lục v và ngược lại.

(b) Ảnh hưởng tiêu cực giữa đỉnh u và các lân cận riêng của nó $N(u)$ được minh họa. Hình tròn màu lục u di chuyển về phía hình vuông màu tím $N(u)$, nhưng nó di chuyển ra xa v.

(c) Ảnh hưởng tích cực giữa đỉnh u và các lân cận riêng của nó $N(u)$ được hiển thị. Hình tròn u màu lục di chuyển về phía hình vuông màu tím $N(u)$. Bởi vì $NN(u)$ được kết nối chặt chẽ với hình vuông màu lục v, v gần u hơn.

Định nghĩa 3.6. (Ảnh hưởng từ các lân cận riêng) Ảnh hưởng của cạnh (u, v) giữa đỉnh u và đỉnh v từ các lân cận riêng được định nghĩa là phương trình 3.8. $d(u, v)$ là Khoảng cách Jaccard địa phương, $\deg(u)$ là bậc của đỉnh u và $f(u)$ là $\sin(u)$. Khoảng cách động đến từ các lân cận riêng $EN(u) = N(u) - \{v\}$ và $EN(v) = N(v) - \{u\}$.

Tương tự như ảnh hưởng từ các đỉnh được liên kết trực tiếp, mỗi lân cận riêng thu hút đỉnh u di chuyển về phía chính nó vì chúng được kết nối. Nhưng chúng ta không biết liệu đỉnh u có xu hướng tiến lại gần đỉnh v hay không. Để minh họa những ảnh hưởng tích cực hoặc tiêu cực của những đỉnh lân cận riêng về khoảng cách, một cách tương tự với cách được đề xuất bởi Shao et al. [13]. Nếu như đỉnh v tương tự với các lân cận riêng của u, việc di chuyển từ u sang các lân cận riêng của nó dẫn đến việc giảm $d(u, v)$. Mặt khác, nếu đỉnh v không tương tự với các lân cận loại trừ của u, thì việc di chuyển từ u sang các lân cận loại trừ của nó dẫn đến sự gia tăng của $d(u, v)$.

$$
\begin{aligned}
EI(u,v) = &- \sum_{x \in EN(u)} \frac{1}{deg(u)} f(1 - d(x,u)) \rho(x,u) \\
&- \sum_{y \in EN(v)} \frac{1}{deg(v)} f(1 - d(y,v)) \rho(y,v)
\end{aligned}
$$

$$
\rho(x,u) = \begin{cases} 1 - d(x,v), (1 - d(x,v)) \geq \lambda \\ 1 - d(x,v) - \lambda, \text{ trong các trường hợp còn lại} \end{cases} \tag{3.8}
$$

Ảnh hưởng tiêu cực của những lân cận riêng được thể hiện trong Hình 7 (b). Vì không có liên hệ nào giữa v và các lân cận bậc hai của u $NN(u)$, $N(u)$ và v không như vậy. Do đó, các lân cận riêng của u thu hút u di chuyển đến gần họ hơn, nhưng u ở xa v.

Ảnh hưởng tích cực của những lân cận riêng được thể hiện trong Hình 7 (c). Do các kết nối giữa v và các lân cận bậc hai của u, $NN(u)$, v và u là tương tự nhau. Các lân cận riêng của u thu hút bạn di chuyển về phía họ và bạn cũng di chuyển đến gần v hơn.

Động lực của việc giới thiệu λ trong biểu thức 3.8 là xác định xem ảnh hưởng của các đỉnh lân cận riêng là tích cực hay tiêu cực. Nếu λ nhỏ, các đỉnh lân cận loại trừ của u có tác động tích cực để làm cho u và v tiến lại gần nhau hơn. Bởi vì các đỉnh lân cận riêng của u tương tự với v, khi λ nhỏ. Nó được biểu thị bằng $\rho \geq 0$. Mặt khác, các đỉnh lân cận loại

trừ của u có tác động ngược chiều làm cho các đỉnh u và v dịch chuyển ra xa. Ảnh hưởng tiêu cực được biểu thị bằng $\rho < 0$.

Định nghĩa 3.7. (Khoảng cách động)

Xem xét các ảnh hưởng từ cả các đỉnh lân cận được kết nối trực tiếp và riêng biệt, chúng tôi giải thích thêm về khái niệm khoảng cách động. Khoảng cách động $d^{t+1}(u, v)$ tại bước thời gian $t + 1$ được xác định bởi khoảng cách $d^{t+1}(u, v)$ tại bước thời gian trước đó t, ảnh hưởng từ tương tác trực tiếp $DI^t(u, v)$ và các lân cận loại trừ $EI^t(u, v)$. Giá trị ban đầu của $d^t(u, v)$ được tính bằng khoảng cách Jaccord địa phương từ biểu thức 3.6. Khoảng cách cập nhật cho đến khi chúng hội tụ ($d^{t+1}(u, v) = 1$ hoặc $d^{t+1}(u, v) = 0$). Khi tất cả các khoảng cách đạt đến trạng thái ổn định, quá trình động chấm dứt. Thuật toán chi tiết có thể được mô tả trong Thuật toán 2 trong phần tiếp theo.

$$d^{t+1}(u, v) = d^t(u, v) + DI^t(u, v) + EI^t(u, v) \tag{3.9}$$

Tiếp theo, chúng tôi minh họa sự giống và khác nhau giữa Attractor và BiAttractor. Khung chung của BiAttractor giống hệt với sơ đồ Attractor ban đầu, bởi vì chúng tôi tuân theo cùng một ý tưởng về động lực. Cả hai đều được thúc đẩy bởi các mô hình tương tác xã hội trong xã hội loài người. Mọi người có xu hướng có mối quan hệ thân thiết với bạn bè của họ trong cùng một cộng đồng, nhưng giữ một khoảng cách nhất định với người lạ. Do đó, khoảng cách giữa các đỉnh trong cùng một cộng đồng là nhỏ nhưng khoảng cách giữa các cộng đồng khác nhau thì lớn. Khi chúng ta mở rộng phương pháp từ mạng đơn thành phần sang mạng hai phần, các phần sau được sửa đổi.

Đầu tiên, Attractor dựa trên Khoảng cách Jaccard để đo khoảng cách động của hai đỉnh bất kỳ trong mạng. Khoảng cách Jaccard dựa trên các đỉnh lân cận chung của hai đỉnh đã cho. Nhưng trong nghiên cứu về các mạng hai phần, chúng ta thấy rằng đỉnh u và đỉnh v là các loại đỉnh khác nhau và các đỉnh lân cận của chúng cũng là các loại đỉnh khác nhau. Do đó, không có đỉnh lân cận chung nào tồn tại về Khoảng cách Jaccard giữa hai loại đỉnh khác nhau bất kỳ trong mạng hai phần. Quan sát thêm cho thấy các lân cận của đỉnh u $N(u)$ có liên kết với các lân cận bậc hai của đỉnh u $NN(u)$. Do đó, chúng tôi đề xuất Khoảng cách Jaccard địa phương (LJD) thay vì khoảng cách Jaccard trong BiAttractor.

Thứ hai, do không có vấn đề về đỉnh lân cận chung trong các mạng hai phần nên ảnh hưởng của đỉnh lân cận chung đối với khoảng cách động bị bỏ qua. Do đó, một điểm khác biệt nữa giữa Attractor và BiAttractor là chỉ ảnh hưởng được liên kết trực tiếp và ảnh hưởng từ các đỉnh lân cận riêng mới được xem xét trong BiAttractor. Nhưng Attractor có ảnh hưởng

của những đỉnh lân cận chung còn BiAttractor thì không có. Hơn nữa, các phương trình về ảnh hưởng được liên kết trực tiếp và ảnh hưởng từ các đỉnh lân cận riêng đến từ phương pháp ban đầu Attractor. Nhưng chúng được sửa đổi để sử dụng Khoảng cách Jaccard địa phương (LJD) thay vì Khoảng cách Jaccard.

Vì hai lý do được trình bày ở trên, các phương trình thay đổi khoảng cách động của Attractor và BiAttractor là khác nhau. Nó được hiển thị trong biểu thức 3.9 để xác định quá trình thay đổi của BiAttractor. $DI^t(u,v)$ và $EI^t(u,v)$ phụ thuộc vào Khoảng cách Jaccard địa phương (LJD).

b. Mẫu

Trong phần trước, các mẫu tương tác được giới thiệu là tiêu chí ban đầu của BiAttractor. Sau đây, chúng tôi sẽ giới thiệu chi tiết về BiAttractor.

Cho một đồ thị hai phía vô hướng và không trọng số $G = (U, V, E)$, trong đó U và V đại diện cho hai tập hợp các loại đỉnh khác nhau. Không có cạnh giữa các đỉnh từ cùng một tập hợp U hoặc V. BiAttractor nhằm mục đích xác định một phân vùng của U và V thành k cộng đồng $C_1, C_2, ..., C_k$ trong đó C_k chứa hai loại đỉnh từ U và V. Mô tả chính thức của thuật toán được minh họa trong Thuật toán 2.

Algorithm 2 Thuật toán BiAttractor

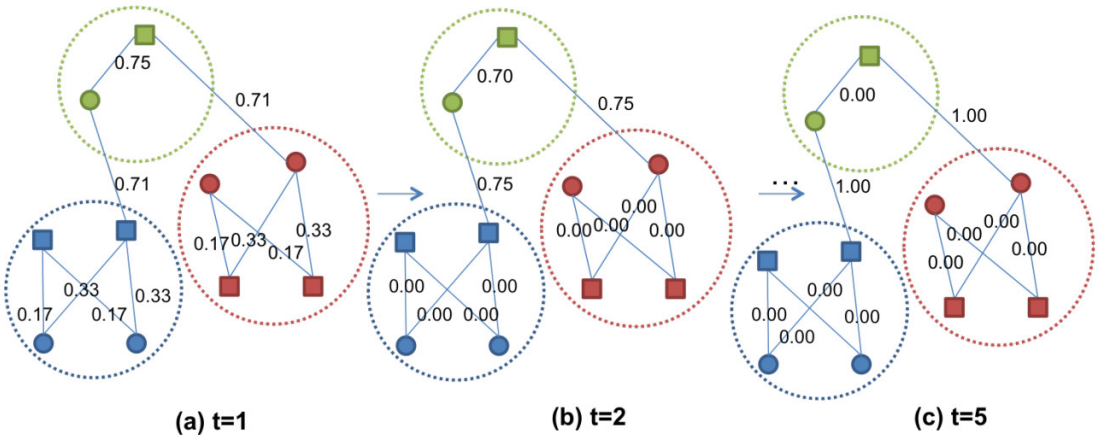
Đầu vào: Cho trước một mạng hai phần vô hướng và không trọng số $G(U, V, E)$ và tham số cố kết λ. λ được cập nhật lặp lại giữa $[0, 1]$ để có được phân vùng cộng đồng tối ưu với Q_b tối đa.
Đầu ra: k cộng đồng hai chế độ $C_1, C_2, ..., C_k$.
Khởi tạo mỗi cạnh $e = \{u, v\}$ từ E với khoảng cách ban đầu $d^0(u, v)$ bởi phương trình 3.6.
Khởi tạo $d^0(u, x)$ trong đó $x \in EN(u)$ bởi phương trình 3.6.
Khởi tạo $d^0(v, y)$ trong đó $y \in EN(v)$ bởi phương trình 3.6.
while không hội tụ từ tất cả các cạnh **do**
    if $0 < d^t(u, v) < 1$ **then**
        $d^{t+1}(u, v) = d^t(u, v) + DI^t(u, v) + EI^t(u, v)$ bởi phương trình 3.9.
    end if
    if $d^t(u, v) <= 0$ **then**
        $d^t(u, v) = 0, d^t$ hội tụ.
    end if
    if $d^t(u, v) >= 1$ **then**
        $d^t(u, v) = 1, d^t$ là hội tụ.
    end if
end while
Tạo các cộng đồng $C_1, C_2, ..., C_k$ bằng cách loại bỏ tất cả các cạnh có $d^t(u, v) = 1$).

a) Ban đầu, mỗi cạnh được gắn với một khoảng cách ban đầu theo khoảng cách Jaccard địa phương theo phương trình 3.6.

b) Theo thời gian, khoảng cách của mỗi cạnh thay đổi do liên kết giữa các đỉnh được kết nối trực tiếp và các lân cận riêng theo phương trình 3.9. Ở đây, kiểu tương tác của BiAttractor khác với kiểu tương tác của Attractor. Các so sánh và lý do đã được giới thiệu trong phần trước về các mẫu tương tác.

c) Khoảng cách động d^t thay đổi lặp đi lặp lại nếu $0 < d^t(u, v) < 1$. Các cạnh trong cùng một cộng đồng có xu hướng giảm và cạnh giữa các cộng đồng khác nhau có xu hướng tăng. Cuối cùng, mọi khoảng cách đều hội tụ ($d^t(u, v) = 0$ hoặc $d^t(u, v) = 1$). Các cạnh được liên kết với giá trị 0 biểu thị các cạnh trong cộng đồng. Những cái được liên kết với giá trị một đại diện cho các cạnh liên cộng đồng đã bị xóa khỏi các mạng bên dưới để hình thành các cộng đồng một cách tự nhiên.

d) Lý do của việc giới thiệu λ là để xác định xem ảnh hưởng của các đỉnh lân cận riêng là tích cực hay tiêu cực. Nếu λ nhỏ, các đỉnh lân cận riêng của u có tác động tích cực để làm cho u và v tiến lại gần nhau hơn. Bởi vì các đỉnh lân cận riêng của u tương tự với v, khi λ nhỏ. Mặt khác, các đỉnh lân cận riêng của u có tác động ngược chiều làm cho các đỉnh u và v dịch chuyển ra xa. λ được cập nhật lặp lại giữa $[0, 1]$ để chạy thuật toán (ví dụ: $\lambda = 0,05$ được đề xuất). Do λ tối ưu, phân vùng cộng đồng tối ưu được chọn với mô đun tối đa Q_b [21].



Hình 8: Một ví dụ của BiAttractor. (a) Đồ thị bao gồm ba thành phần phụ theo các màu khác nhau trong đó hình tròn và hình vuông biểu thị hai loại đỉnh khác nhau. Khoảng cách gắn vào các cạnh được khởi tạo ở bước đầu tiên. (b) Ở bước thời gian thứ hai, khoảng cách được cập nhật theo

BiAttractor. (c) Cuối cùng, tất cả các cạnh trong cộng đồng có độ lớn 0 và các cạnh giữa các cộng đồng có độ lớn bằng một. Loại bỏ các cạnh được liên kết với giá trị có thể tự động phát hiện các cộng đồng.

Để hiểu rõ hơn về thuật toán BiAttractor, ý tưởng cơ bản được trình bày ngắn gọn bằng một ví dụ trong Hình 8. Cho một đồ thị hai phần đơn giản có mười đỉnh và mười một cạnh, hình tròn và hình vuông đại diện cho hai loại đỉnh riêng biệt. Khoảng cách cạnh biểu thị mối quan hệ giữa hai đỉnh được kết nối bởi cạnh. Các giá trị nhỏ hơn luôn biểu thị rằng chúng có nhiều khả năng ở trong cùng một cộng đồng hơn. Ở bước đầu tiên, mỗi cạnh được cung cấp một khoảng cách ban đầu theo các cấu trúc tô pô địa phương được tính bởi phương trình 3.6 như trong Hình 8 (a). Khoảng cách sẽ được cập nhật bởi phương trình 3.9 nếu chúng không đạt đến trạng thái ổn định như trong Hình 8 (b). Sau khi tất cả các khoảng cách hội tụ về 0 hoặc 1, các cộng đồng có thể được phân vùng khỏi mạng bên dưới bằng cách loại bỏ các cạnh có khoảng cách 1 như trong Hình 8 (c). Được biết, ba cộng đồng được phát hiện bằng cách loại bỏ hai cạnh liên cộng đồng.

c. Độ phức tạp

Để nghiên cứu độ phức tạp về thời gian của BiAttractor, mỗi cạnh của mạng bên dưới đã được tính toán một lần để thu được giá trị ban đầu. Như được hiển thị trong dòng 3 trong thuật toán 2, do đó thời gian $O(|E|)$ được xem xét ở bước đầu tiên. Tiếp theo, như được hiển thị trong dòng 4 và dòng 5, các cấu trúc tô pô địa phương yêu cầu BiAttractor thu được ảnh hưởng từ các lân cận riêng. Độ phức tạp thời gian ít nhất là $O(k|E|)$ trong đó k là số lân cận riêng trung bình. Nó được hiển thị từ dòng 6 đến dòng 16 sau T bước thời gian, BiAttractor hội tụ đến trạng thái cuối cùng. Độ phức tạp là $O(T|E|)$ trong quá trình lặp.

Do đó, tổng độ phức tạp thời gian của BiAttractor là $O(|E|+k|E|+T|E|)$. T là hằng số trong thử nghiệm của chúng tôi với mạng bên dưới G. Độ phức tạp thời gian $O(|E|+k|E|+T|E|)$ có thể được viết là $O(z|E|)$ trong đó z là nhãn thông thường của mạng độ trung bình. Nếu mạng dày đặc, độ phức tạp ít nhất là bậc hai. Mạng thực thường là mạng thưa thớt và z không đổi. Độ phức tạp thời gian là tuyến tính trong trường hợp này.

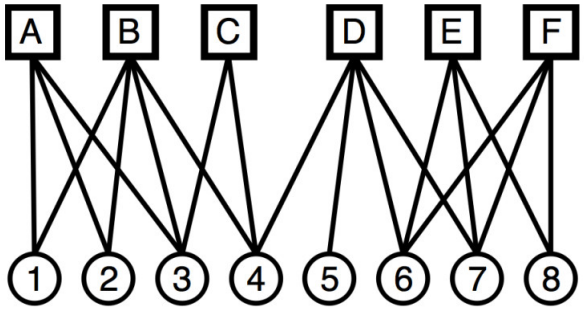
3.2. Thuật toán phát hiện cộng đồng trên mạng hai phần: ComSim

3.2.1. Hàm tương tự

Trong phần này, chúng tôi chính thức trình bày thuật toán phát hiện dành cho đồ thị hai phía. Đầu tiên chúng tôi nhắc lại các định nghĩa cần thiết (Phần 3.2.1) trước khi mô tả

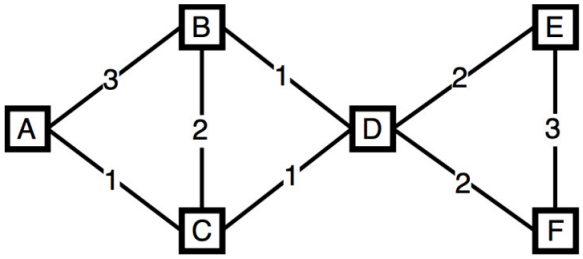
thuật toán ComSim (Phần 3.2.2).

Một đồ thị hai phần xác định bởi: $G = (U, V, E)$ trong đó U là tập hợp các đỉnh trên, V là các đỉnh ở phía dưới (ví dụ diễn viên) và $E \subseteq U \times V$ tập hợp các cạnh viên kết giữa U và V (chẳng hạn là các liên kết giữa diễn viên và các bộ phim họ tham gia).



Hình 9: Ví dụ của một đồ thị hai phần $G = (U, V, E)$

So với biểu đồ một phần, các đỉnh trong biểu đồ hai phần được tách thành hai tập hợp rời rạc và các liên kết luôn nằm giữa một đỉnh trong một tập hợp và một đỉnh trong tập hợp kia. Nhưng điều tự nhiên là cũng phải điều tra xem các đỉnh trong cùng một tập hợp có quan hệ với nhau như thế nào. Cách tiếp cận này thường được nắm bắt bởi khái niệm hình chiếu của một đồ thị hai phần trên một trong hai tập hợp của nó.



Hình 10

Ví dụ về phép chiếu U có trọng số của G sử dụng đỉnh lân cận chung làm hàm tương tự

Chẳng hạn, nếu quan tâm đến phép chiếu U, người ta có thể nghiên cứu cách các đỉnh U kết nối theo độ tương tự của chúng được đo bằng các liên kết của chúng đối với các đỉnh U chung. Về mặt hình thức, độ tương tự như vậy được nắm bắt bởi hàm tương tự θ. Điều này cho phép xác định chính thức đồ thị được chiếu theo trọng số $G_U = (U; \theta)$ trong đó $\theta : U \times U \mapsto R^+$. Do đó, biểu đồ này chỉ ra sức mạnh của mối quan hệ giữa các đỉnh U. Do đó, phép chiếu U của biểu đồ hai bên trong Hình 9 sẽ dẫn đến biểu đồ được mô tả trong Hình 10.

Chú ý rằng trong phần còn lại của 3.2, chúng ta sẽ sử dụng hàm lân cận chung tiêu chuẩn $\theta(x, y) = |N(x) \cap N(y)|$. Nhưng cách tiếp cận này dễ dàng mở rộng sang các hàm số tương tự khác như chỉ số jaccard [22], phân bổ các đỉnh [22] hoặc hệ số adamic-adar [23]

3.2.2. Thuật toán COMSIM

Cho một đồ thị hai phía $G = (U, V, E)$ và một hàm tương tự θ chẳng hạn như các đỉnh lân cận chung, COMSIM tạo một phân vùng theo hai bước. Đầu tiên, nó xác định các cộng đồng cốt lõi, đó là các nhóm đỉnh có độ tương tự cao theo hàm θ.

Như mọi người có thể thấy trong Thuật toán 3 trình bày chi tiết bước đầu tiên này, thuật toán tạo ra một chuỗi các đỉnh bằng cách đi theo các liên kết ra ngoài có trọng số cao nhất theo θ. Khi chuỗi này đến một đỉnh đã được xem xét, điều đó có nghĩa là một chu kỳ đã được phát hiện trong chuỗi. Chu kỳ này sau đó tạo thành cốt lõi của một cộng đồng trong tương lai.

Trên ví dụ về trong Hình 9, nó sẽ dẫn đến việc phát hiện ra rằng các đỉnh A và B tạo thành cốt lõi của một cộng đồng, cũng như E và F. Điều này phù hợp với Hình 10 cho thấy rằng A và B, cũng như E và F có các liên kết có trọng số cao nhất. Các đỉnh khác (C và D) được để lại trong tập K còn lại.

Sau đó, giai đoạn thứ hai của thuật toán cố gắng định vị các đỉnh còn lại của K trong các cộng đồng hiện có bằng cách tối đa hóa độ tương tự giữa các đỉnh này và tất cả các đỉnh của các cộng đồng cốt lõi.

Như được mô tả trong Thuật toán 4, bước thứ hai xem xét tất cả các đỉnh còn lại không phải là một phần của phân vùng sau bước đầu tiên. Đối với mỗi nút x, nó xác định các cộng đồng có ít nhất một liên kết với x. Sau đó, thuật toán sẽ chọn cộng đồng tối đa hóa tổng số điểm tương đồng giữa x và tất cả các đỉnh của cộng đồng.

Trên ví dụ về đồ chơi trong Hình 9, và không phụ thuộc vào thứ tự mà các đỉnh C và D được xem xét trong bước 2, nó sẽ dẫn đến việc ảnh hưởng từ đỉnh C đến cộng đồng $A - B$ (tổng các điểm tương đồng là 3) và đỉnh D đến cộng đồng $E - F$ (tổng các điểm giống nhau là 4).

Điều đáng chú ý là vì một số liên kết có thể có trọng số tương tự nhau nên hai bước có thể phải đối mặt với một số tùy chọn bằng nhau. Trong trường hợp đó, thuật toán sẽ chọn ngẫu nhiên một tùy chọn thống nhất trong số tất cả các tùy chọn có thể. Vì lý do này, thuật toán không xác định và một số lần chạy có thể dẫn đến các phân vùng khác nhau.

Thuật toán phát hiện cộng đồng trên mạng hai phần sử dụng tính tương tự của chu trình và đỉnh.

Algorithm 3 Thuật toán COMSIM - Bước đầu tiên

Đầu vào: Một đồ thị hai phần $G = (U, V, E)$, hàm tương tự θ
Đầu ra: trả về một phân vùng P gồm các đỉnh U và một tập K các nút còn lại (cho bước thứ hai)
$P := \emptyset$ // tập hợp phân vùng
$T := U$ //tập hợp các đỉnh được xem xét
$x := rand\_and\_remove(T)$ // đỉnh ngẫu nhiên
$H := \emptyset$
$K := \emptyset$ //tập hợp các đỉnh còn lại
while $T \neq \emptyset$ **do**
    tìm một đỉnh lân cận $y \in N(N(x))$ của x mà hàm $\theta(x, y)$ đạt giá trị lớn nhất.
    $y := argmax_{y \in N(N(x))} \theta(x, y)$
    if $y \in H$ **then**
        $C := cycle(H, y, x)$ //trích xuất chu trình được phát hiện từ y đến x trong H.
        P.add(C)
        $K := K \cup (H - C)$ //lưu trữ các đỉnh không có trong chu trình C
        $H := \emptyset$
        $x := rand\_and\_remove(T)$
    end if
    if $y \notin H$ **then**
        if $y \in T$ **then**
            $H := H \cup \{y\}$
            $x := y$
            $T := T - \{y\}$
        end if
        if $y \notin T$ **then**
            y đã là một phần tử của P, các đỉnh đã truy cập được lưu trữ
            $K = K \cup H$
            $H := \emptyset$
            $x := rand\_and\_remove(T)$
        end if
    end if
end while
return P and K

Algorithm 4 Thuật toán COMSIM - Bước thứ hai

Đầu vào: Một đồ thị hai phần $G = (U, V, E)$, một phân vùng P, Một tập hợp K các đỉnh dư lại sau (bước đầu tiên) hàm tương tự θ
Đầu ra: trả về một phân vùng P' gồm các đỉnh của U và các đỉnh không thỏa mãn R
$R := \emptyset$
$P' := P$
for với mỗi $x \in K$ **do**
    $P_x = com_neigh(x, P)$ // Tìm tất cả các cộng đồng lân cận của x
    if $P_x = \emptyset$ **then**
        $R := R \cup \{x\}$
    end if
    if $P_x \neq \emptyset$ **then**
        $C := argmax_{C_x \in P_x} \sum_{y \in C_x} \theta(x, y)$
        Thêm x vào phân vùng C của P'
    end if
end for
return P' và R

3.2.3. Cải tiến thuật toán COMSIM

Sau khi phân cụm được một bên của đồ thị hai phần ban đầu thành các cộng đồng chúng tôi tiếp tục phân cụm cho bên còn lại. Từ trên chúng tôi đã có các cụm cộng đồng ở trong tập hợp đỉnh U với các đỉnh thuộc tập hợp đỉnh V chúng tôi xét số liên kết từ mỗi đỉnh đến các cộng đồng trong U số liên kết với cộng đồng nào nhiều nhất ta sẽ ghép điểm vào cộng đồng đó.

Ví dụ ở phía trên ta đã phân cụm được tập hợp đỉnh U thành hai cộng đồng gồm đỉnh A, B, C và cộng đồng gồm đỉnh D, E, F. Ta xét các điểm thuộc V ví dụ điểm 1. Ta thấy tổng các liên kết từ 1 đến cộng đồng A, B, C là 2 và 1 không có liên kết với cộng đồng D, E, F vậy 1 được ghép vào cộng đồng gồm 3 đỉnh A, B, C.

Cuối cùng ta sẽ thu được cộng đồng cần tìm.

Chương 4
MỘT SỐ THÍ NGHIỆM

4.1. So sánh thuật toán Attractor với một số thuật toán khác

Trong phần này, chúng tôi đánh giá thuật toán Attractor trên các mạng tổng hợp cũng như trong thế giới thực để chỉ ra hiệu quả của nó. Trong phần này chúng tôi đã tham khảo kết quả so sánh trong tài liệu [13] đồng thời chạy thử nghiệm trên python (phụ lục A).

Lựa chọn phương pháp so sánh. Để đánh giá hiệu suất của Attractor, chúng tôi so sánh nó với một số đại diện của thuật toán phát hiện cộng đồng.

Ncut [24] là một thuật toán nổi tiếng để phân cụm đồ thị bằng cách tối ưu hóa tiêu chí cắt chuẩn hóa. Vì sự phân tích giá trị riêng được áp dụng để tăng tốc độ tìm ra vết cắt tối ưu, nên nó cũng thường được gọi là phân cụm quang phổ.

Modularity [25] là thuật toán phát hiện cộng đồng phổ biến nhất hiện nay dựa trên phép đo modulairty, sử dụng phần cắt dự kiến để đo chất lượng phân cụm.

Metis [26] là một cách tiếp cận phân cụm đồ thị rất nhanh cho các mạng lớn thông qua phân vùng đa cấp và thực hiện song song.

MCL [27] là một thuật toán phổ biến được sử dụng trong khoa học đời sống dựa trên mô phỏng các luồng (ngẫu nhiên) trong đồ thị.

Louvain [28] là một thuật toán dựa trên mô đun nổi tiếng khác. So với thuật toán Modularity do Newman [11] đề xuất, nó cho phép phát hiện cộng đồng có thứ bậc và có độ phức tạp về thời gian thấp hơn.

Infomap [29] coi vấn đề phát hiện cộng đồng là một vấn đề mã hóa và nhằm mục đích tìm các phân vùng tối ưu dựa trên nguyên tắc độ dài mô tả tối thiểu.

Đối với Louvain và Infomap, các tham số mặc định được sử dụng. Chúng tôi đặt tham số gắn kết $\lambda = 0,5$ cho Attractor làm tham số mặc định.

Ma trận đánh giá. Để so sánh rộng rãi các thuật toán phát hiện cộng đồng khác nhau về hiệu quả, chúng tôi đánh giá kết quả phân cụm theo hai cách.

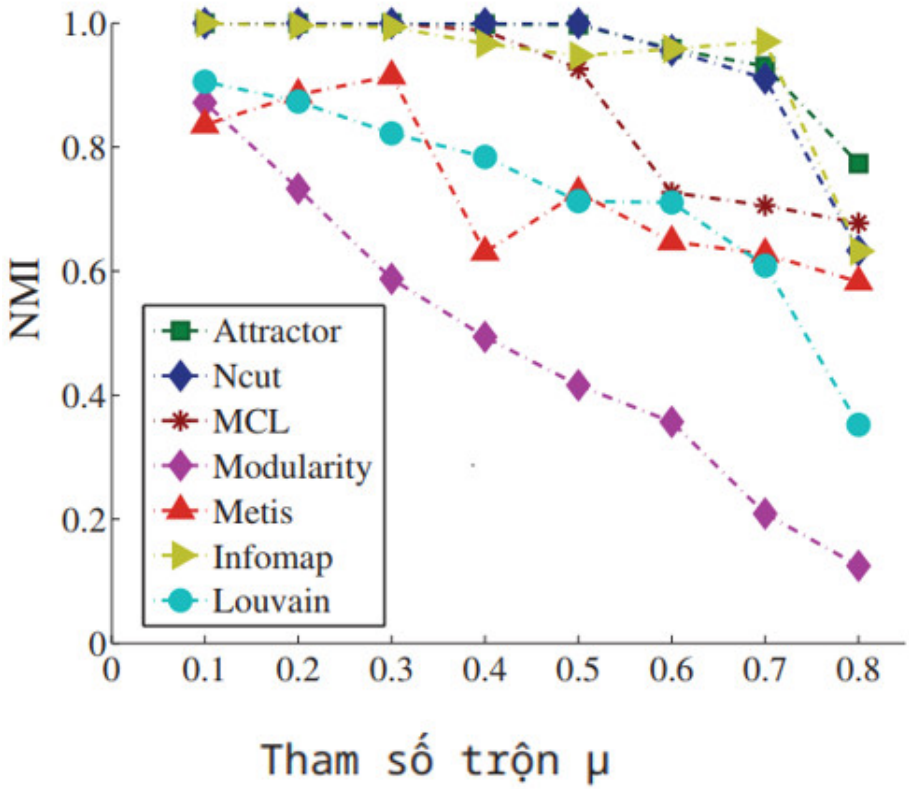
Mạng có nhãn lớp. Đối với các mạng có cộng đồng đã được biết đến, hiệu suất được đo trực tiếp bằng ba biện pháp đánh giá được sử dụng rộng rãi: Thông tin tương hỗ được chuẩn hóa (NMI) [30], Chỉ số Rand được điều chỉnh (ARI) [31] và Độ tinh khiết của cụm.

Mạng không có nhãn lớp. Vì sự thật cơ bản của cấu trúc cộng đồng vẫn chưa được biết, nên việc so sánh hiệu suất của các thuật toán riêng biệt theo một cách khách quan là một nhiệm vụ không hề nhỏ. Để đánh giá chất lượng của các cộng đồng được tạo ra bởi các thuật toán khác nhau một cách hợp lý, hai phép đo nội bộ phổ biến là tính mô-đun [25] và phép cắt chuẩn hóa (ncut) [24] đã được áp dụng trong nghiên cứu này.

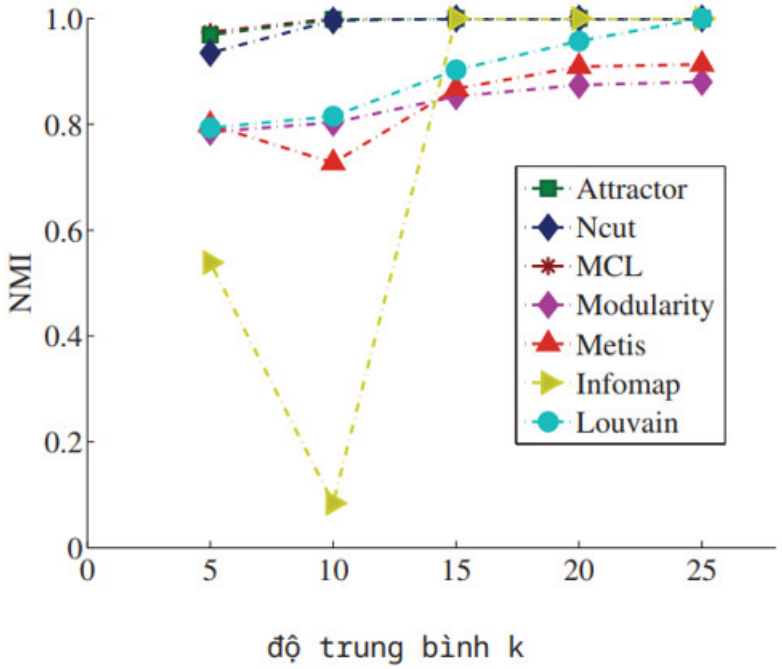
4.1.1. Mạng tổng hợp

Trong phần này, trước tiên chúng tôi tạo một số mạng tổng hợp có các đặc điểm riêng biệt để so sánh hiệu suất của các thuật toán phát hiện cộng đồng khác nhau. Để so sánh công bằng và để làm cho các mạng tổng hợp nhất quán hơn với các mạng trong thế giới thực, các mạng chuẩn LFR [32] đã được áp dụng, trong đó có thể dễ dàng kiểm soát sự phân phối mức độ và quy mô cộng đồng của các mạng. Để tăng độ phức tạp của mạng, tham số trộn μ [32], được định nghĩa là tỷ lệ liên kết của mỗi đỉnh bên ngoài cộng đồng của nó, được sử dụng để kiểm soát độ khó của việc tách cộng đồng.

***Noise Edge**:* Trước tiên, chúng tôi đánh giá mức độ hiệu quả của các thuật toán phân cụm biểu đồ khác nhau cho phép phát hiện các cộng đồng bằng cách thay đổi các cạnh liên cụm của chúng. Các cạnh liên cụm, mà chúng tôi gọi là các cạnh nhiễu, được thêm vào mạng để cản trở sự phân tách cộng đồng. Chúng tôi sửa mức độ trung bình của nút và kích thước cộng đồng, đồng thời thay đổi tham số trộn μ từ $0,1$ thành $0,8$ để tạo ra một chuỗi các mạng có các cạnh liên cụm khác nhau. Tất cả các mạng bao gồm 2000 nút với mức độ trung bình $k = 20$.



Hình 11: Hiệu suất của các thuật toán khác nhau trong LFR chuẩn mạng bằng cách thay đổi số cạnh giữa các cụm.



Hình 12: Hiệu suất của các thuật toán khác nhau trên mạng điểm chuẩn LFR bằng cách thay đổi mật độ cộng đồng bằng mức độ trung bình $< k >$.

Data Sets	|V|	|E|	#Class	AD	CC
Zarachy	34	78	2	4.588	0.571
Football	115	613	11	10.661	0.403
Polbooks	105	441	3	8.400	0.488
Amazon	334863	925872	151037	5.530	0.397
Collaboration	9875	25973	-	5.260	0.312
Friendship	58228	214078	-	7.353	0.172
Road	1088092	1541898	-	2.834	0.047

Bảng 1: Thống kê các tập dữ liệu thực tế, trong đó AD: trình độ trung bình; CC: hệ số phân cụm.

Với việc tăng tham số trộn, hiệu suất (được đo bằng NMI) của cả năm phương pháp được hiển thị trong Hình 11. Chúng ta có thể thấy rằng các thuật toán của Attractor, Ncut và MCL gần như đạt được các cụm hoàn hảo bằng cách thêm các cạnh giữa các cụm với tham số trộn lên tới $0,4$ (40% cạnh của mỗi đỉnh liên kết với các cộng đồng khác).

Hiệu suất của chúng bắt đầu giảm khi ngày càng có nhiều cạnh xen kẽ được thêm vào mạng và Attractor mạnh hơn đối với các cạnh nhiễu này. Đối với Modularity và Louvain, chúng nhạy cảm hơn với các cạnh nhiễu này và hiệu suất của chúng không thể so sánh với năm thuật toán khác trên các mạng này. Đối với thuật toán Metis, hiệu suất của nó bị dao động và bắt đầu giảm đáng kể ngay khi thêm nhiều cạnh xen kẽ (với $\mu = 0,4$). Hiệu suất của infomap là đáng ngạc nhiên, vì hiệu suất của nó giảm đầu tiên và sau đó tăng lên với μ nằm trong khoảng từ 0,5 đến 0,7.

Community Density: Tiếp theo, chúng tôi đánh giá cách các thuật toán phản ứng với các mạng có mức độ trung bình khác nhau, mà chúng tôi gọi là mật độ cộng đồng. Ở đây, chúng tôi sửa các cạnh liên cụm ($\mu = 0,1$) và thay đổi độ trung bình k từ 5 thành 25 để xem ảnh hưởng của mật độ cộng đồng đến hiệu suất của các thuật toán này. Hình 6 cho thấy Attractor, MCL và Ncut mang lại kết quả tốt cho tất cả các mạng này, trong khi hiệu suất của Metis và Modularity kém hơn một chút. Chúng ta có thể thấy rằng Attractor, Metis và Ncut cho phép tìm kiếm chính xác các cộng đồng tốt ngay cả với mật độ cộng đồng thấp ($k = 5$). Đối với Louvain, Metis và Modularity, chúng nhạy cảm hơn với mật độ cộng đồng trên các mạng tổng hợp này. Đối với Infomap, nó hiển thị hiệu suất bất thường khi bậc k trung bình là 10.

4.1.2. Dữ liệu thực

Trong phần này, chúng tôi đánh giá hiệu suất của các thuật toán phát hiện cộng đồng khác nhau trên các mạng trong thế giới thực, tất cả đều có sẵn công khai từ kho lưu trữ dữ

liệu mạng UCI (https://networkdata.ics.uci. edu/index.php) và mạng lớn Stanford bộ sưu tập dữ liệu (http://snap.stanf ord.edu/data/). Số liệu thống kê của bảy mạng được tóm tắt trong Bảng 1.

	Zarachy			Football			Polbooks			Amazon		
	NMI	ARI	Pur.	NMI	ARI	Pur.	NMI	ARI	Pur.	NMI	ARI	Pur.
Attractor	**0.859**	**0.939**	**1.000**	**0.923**	**0.897**	**0.930**	**0.559**	**0.680**	**0.857**	**0.931**	**0.580**	**0.998**
Ncut	0.833	0.882	0.970	**0.923**	**0.897**	**0.930**	0.534	0.645	0.829	–	–	–
Modulairty	0.577	0.680	0.970	0.596	0.474	0.574	0.508	0.638	0.838	–	–	–
Metis	0.836	0.882	0.970	0.393	0.095	0.339	0.502	0.516	0.781	0.761	0.092	0.989
MCL	0.833	0.882	0.970	**0.923**	**0.897**	**0.930**	0.455	0.594	**0.857**	0.902	0.490	0.991
Louvain	0.524	0.541	**1.000**	0.858	0.807	0.870	0.440	0.537	**0.857**	0.738	0.384	0.384
Infomap	0.593	0.702	0.971	0.906	0.857	0.904	0.476	0.646	0.848	0.209	0.009	0.077

Bảng 2: Hiệu suất của các thuật toán phân cụm đồ thị khác nhau trên các mạng thế giới thực được gắn nhãn.

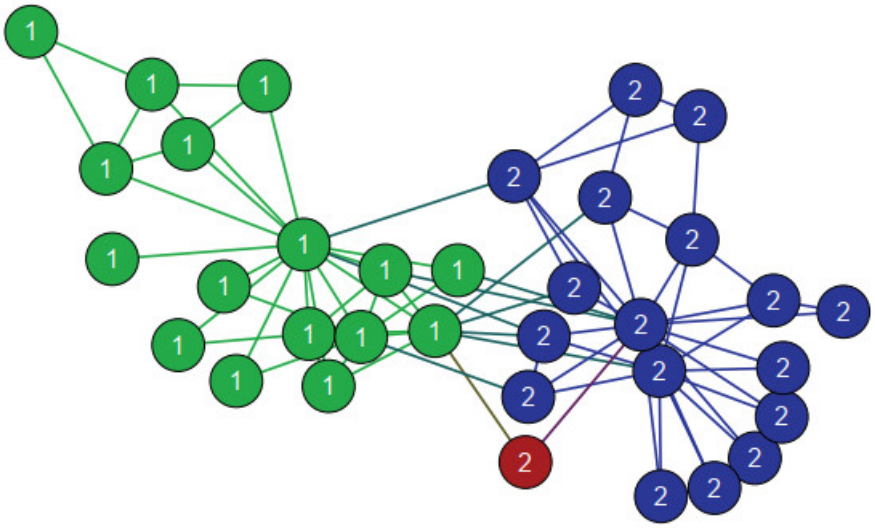
	Collaboration			Friendship			Amazon			Road		
	#C	*mod.*	*ncut*	#C	*mod.*	*ncut*	#C	*mod.*	*ncut*	#C	*mod.*	*ncut*
Attractor	1384	0.579	1179	8045	0.421	7325	23825	0.741	10811	59919	0.856	25055
Metis	1384	0.309	4217	8045	0.138	53984	23825	0.451	47336	59919	0.673	31542
MCL	2093	0.537	2103	13788	0.319	36723	46557	0.623	47488	86745	0.810	25065
Louvain	475	0.768	10.120	746	0.684	38.340	240	0.926	9.617	492	0.989	2.032
Infomap	456	0.722	5.470	572	0.439	4.104	12	0.4224	0.1249	208	0.660	6.088

Bảng 3: Hiệu suất của các thuật toán khác nhau trên các mạng thế giới thực lớn không có thông tin về lớp.

(1). Mạng có thông tin các lớp Trước tiên, chúng tôi điều tra các mạng mà sự thật cơ bản về cấu trúc cộng đồng đã được biết đến. Các biện pháp bên ngoài như NMI, ARI và độ tinh khiết được báo cáo.

Mạng lưới câu lạc bộ karate của Zachary: Mạng lưới nổi tiếng, bắt nguồn từ quan sát của Zachary về một câu lạc bộ karate, phản ánh mối quan hệ bạn bè giữa các thành viên này. Đặc biệt, mạng có thể được chia thành hai cộng đồng, điều này phản ánh sự bất đồng giữa quản trị viên và người hướng dẫn. Hình 13 cho thấy Attractor xác định các cộng đồng có mức độ thành công cao (với giá trị NMI, ARI và Độ tinh khiết cao) và vượt trội hơn các thuật toán so sánh khác (Bảng 2). Cụ thể, hai cộng đồng được tìm thấy thành công, ngoại trừ một thành viên được xem là nhiễu (đỉnh 10). Cũng thật thú vị khi quan sát thấy thành viên này nằm giữa hai cộng đồng và liên kết tương ứng với các đỉnh trung tâm của hai cộng đồng. Trong kịch bản thế giới thực, rất khó để xác định cộng đồng của nó thuộc về. Trên thực tế, có nhiều khả năng sẽ gán đỉnh này cho cả hai cộng đồng, đây là cách phân cụm chồng chéo mà chúng tôi sẽ không thảo luận ở đây. Để so sánh các thuật toán của MCL và Ncut, chúng cũng đạt được hiệu suất tốt và hầu hết các thành viên đều được nhóm chính xác. Tuy nhiên, đối với các thuật toán Moduleity, Louvain và Infomap, nhiều thành viên bị

nhóm sai, dẫn đến giá trị NMI tương đối thấp. Hiệu suất của các thuật toán khác nhau được tóm tắt trong Bảng 2.

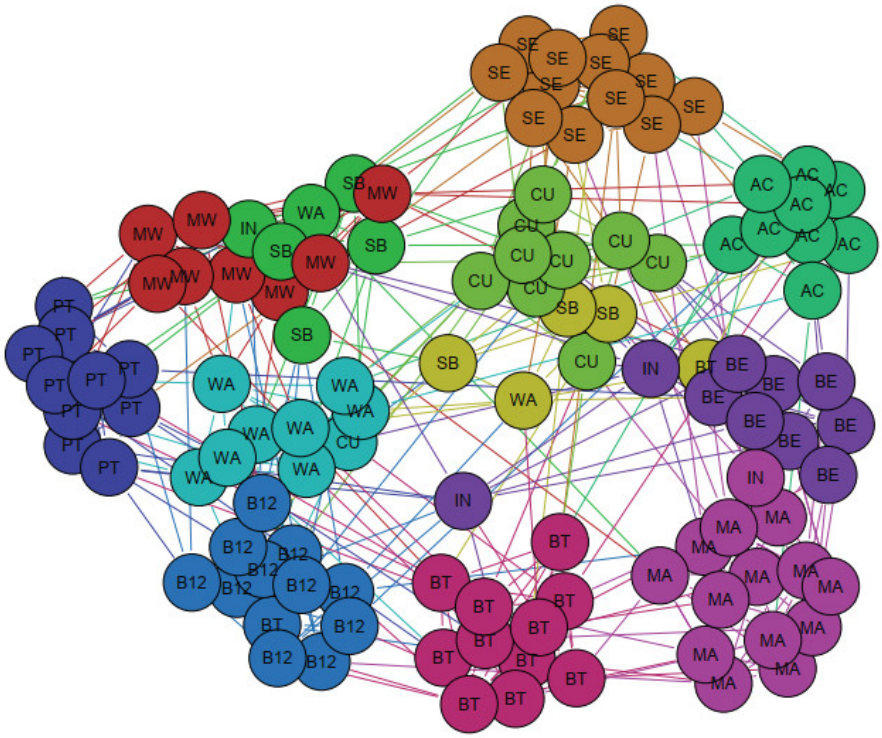


Hình 13: Attractor trên mạng lưới câu lạc bộ karate. Màu sắc của các đỉnh chỉ ra các cộng đồng được phát hiện khác nhau.

Mạng Amazon*:* Mạng này bao gồm 334.863 đỉnh và 925.872 cạnh và mỗi đỉnh đại diện cho một sản phẩm trên trang web Amazon. Mỗi sản phẩm được phân loại theo cộng đồng tương ứng dựa trên danh mục do Amazon cung cấp và 5.000 cộng đồng hàng đầu có chất lượng cao nhất đã được điều tra trong [33]. Do độ phức tạp về thời gian và không gian cao của quá trình phân tách giá trị riêng trong Ncut và Modularity, chúng không thể xử lý mạng này. Dựa vào một phần cộng đồng dựa trên sự thật cơ bản (5.000 cộng đồng hàng đầu), Attractor đạt được chất lượng cộng đồng tốt nhất so với các thuật toán khác có số đo cao (NMI= $0,931$, ARI = $0,580$, Độ tinh khiết = $0,998$) (Bảng 2). Thuật toán MCL cho phép đưa ra một kết quả hợp lý (NMI = 0,902). Tuy nhiên, đối với Metis, Louvain và Infomap, chúng có xu hướng thất bại, đặc biệt là đối với thuật toán Infomap (NMI = 0,209). Ngoài ra, để đánh giá toàn diện, tất cả các kết quả của 5 thuật toán còn được đánh giá bằng các tiêu chí nội tại về mô đun và ncut (xem Bảng 3).

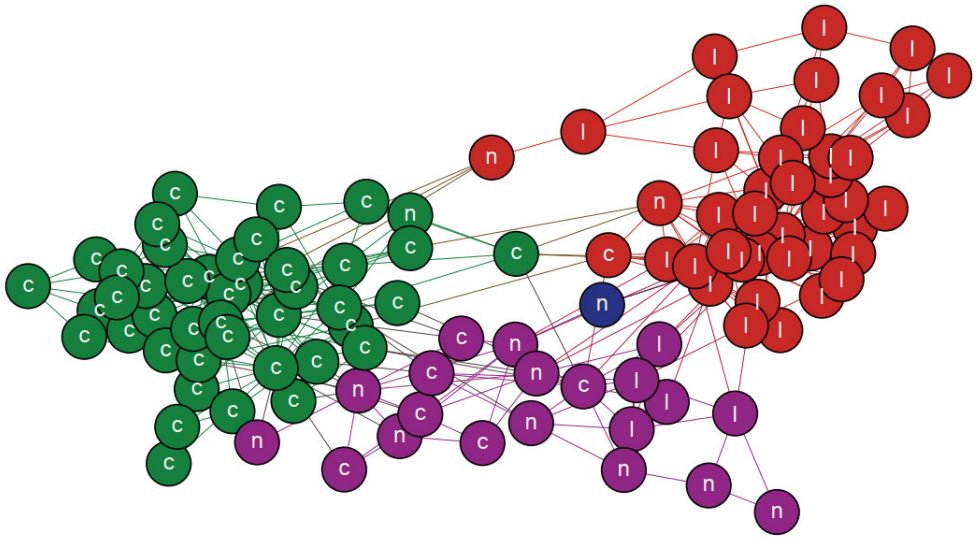
Bóng đá đại học Mỹ (American college football): Mạng bắt nguồn từ các trận đấu bóng đá Mỹ theo lịch thi đấu của Division I trong mùa giải thông thường Mùa thu năm 2000, trong đó 115 đỉnh trong đồ thị đại diện cho các đội và 613 cạnh đại diện cho các trận đấu thông thường trong mùa giải giữa hai đội mà chúng kết nối. Các đội được chia thành 12 hội nghị, mỗi hội nghị có khoảng 8-12 đội và do đó cấu trúc cộng đồng thực sự đã được biết đến. Hình 14 vẽ sơ đồ các cộng đồng được Attractor phát hiện. Điều thú vị cần lưu ý

là Attractor tự động tìm ra 12 cộng đồng có chất lượng cao (NMI = 0,923, ARI = 0,897, Purity =93, 0%). Từ Hình này, chúng ta có thể quan sát thấy hầu hết các nhóm được phân công chính xác vào các cộng đồng tương ứng. Ncut và MCL nhận thấy cấu trúc cộng đồng tương tự như Attractor. Tuy nhiên, đối với Metis, Modulity, Louvain và Infomap, chúng rất khó khám phá cấu trúc cộng đồng tự nhiên (Bảng 2).



Hình 14: Attractor trên mạng bóng đá Mỹ.

Sách về chính trị Hoa Kỳ (Books about US politics): Mạng lưới này, bắt nguồn từ những cuốn sách chính trị về chính trị Hoa Kỳ được xuất bản vào khoảng thời gian diễn ra cuộc bầu cử tổng thống năm 2004, bao gồm 105 đỉnh và 441 cạnh. Các đỉnh đại diện cho sách được bán bởi nhà bán sách trực tuyến Amazon.com. Các cạnh thể hiện việc cùng mua sách thường xuyên bởi cùng một người mua. Mỗi cuốn sách được gắn nhãn 'l', 'n' hoặc 'c' để cho biết chúng là "tự do", "trung lập" hay "bảo thủ", dựa trên việc Newman đọc mô tả và đánh giá về các cuốn sách được đăng trên Amazon . Thuật toán Attractor cho phép phân nhóm tốt những cuốn sách này thành các danh mục, trong đó hai cụm tương ứng thể hiện rõ ràng các cuốn sách tự do và bảo thủ tương ứng (Hình 15 và Bảng 2). Đối với các thuật toán Mô-đun, Metis và Ncut, chúng mang lại kết quả tương đương, trong khi MCL và Infomap tạo ra một nhóm tương đối kém trên mạng này.



Hình 15: Attractor trên trên mạng sách chính trị Hoa Kì.

Mạng Amaron: Mạng này bao gồm 334.863 đỉnh và 925.872 cạnh và mỗi đỉnh đại diện cho một sản phẩm trên trang web Amazon. Mỗi sản phẩm được phân loại theo cộng đồng tương ứng dựa trên danh mục do Amazon cung cấp và 5.000 cộng đồng hàng đầu có chất lượng cao nhất đã được điều tra trong [33]. Do độ phức tạp cao về thời gian và không gian của việc phân tách giá trị riêng trong Ncut và Mô-đun, chúng không thể xử lý mạng này. Dựa vào một phần cộng đồng sự thật cơ bản (5.000 cộng đồng hàng đầu), Attractor đạt được chất lượng cộng đồng tốt nhất so với các thuật toán khác có số đo cao (NMI = 0,931, ARI = 0,580, Độ tinh khiết = 0,998) (Bảng 2). Thuật toán MCL cho phép đưa ra kết quả hợp lý (NMI = 0,902). Tuy nhiên, đối với Metis, Louvain và Infomap, chúng có xu hướng thất bại, đặc biệt đối với thuật toán Infomap (NMI = 0,209). Hơn nữa, để đánh giá toàn diện, tất cả kết quả của 5 thuật toán còn được đánh giá bằng tiêu chí nội bộ module hóa và ncut (xem Bảng 3).

(2). Mạng không có thông tin lớp

Trong phần này, do độ phức tạp về thời gian của Ncut và Modularity, chúng tôi giới hạn việc so sánh với các thuật toán phân cụm Metis, MCL, Louvain và Infomap trên các mạng quy mô lớn không có sự thật về lớp (Bảng 3). Vì không có biện pháp thuyết phục nào cho mạng không được gắn nhãn, chúng tôi sử dụng Modularity và Ncut để đánh giá các thuật toán này theo cách cung cấp thông tin.

Mạng cộng tác Hepth (Hepth collaboration network): Mạng này là mạng cộng tác gồm 9.875 tác giả làm việc về lý thuyết vật lý năng lượng cao. Thuật toán Attractor xác định 1384 cộng đồng, kết quả là $modularity = 0,579$ và $ncut = 1179$. Trên tập dữ liệu,

Metis ($K = 1384$) và MCL cũng mang lại khả năng phân vùng tốt trong khi hiệu suất của nó kém hơn Attractor xét theo hai thước đo (Bảng 3). Nếu chúng ta chỉ nhìn vào mô-đun và ncut, thì Louvain và Infomap có vẻ tốt hơn nhiều so với Attractor, MCL và Metis. Tuy nhiên, lý do là hai thuật toán chỉ mang lại một vài cộng đồng, tự nhiên dẫn đến các giá trị mô đun và ncut tốt hơn, dựa trên các định nghĩa. Nếu chúng tôi tác động số lượng cộng đồng khác nhau, thì Attractor thực sự đạt được hiệu suất tốt hơn.

Mạng tình bạn Brightkite: Biểu đồ là mạng tình bạn dựa trên vị trí bao gồm 58.228 đỉnh và 214.078 cạnh vô hướng. Attractor tìm thấy 8045 cộng đồng và cho thấy lợi thế rõ ràng so với hai thuật toán khác dựa trên hai thước đo. Đối với Metis ($K = 8045$), nhiều người bạn dường như được nhóm không chính xác, dẫn đến giá trị mô đun thấp $= 0,138$. Như thường lệ, Louvain và Infomap, hai thuật toán có xu hướng tạo ra số lượng cộng đồng nhỏ và không thể phát hiện được nhiều cộng đồng có quy mô nhỏ. Nguyên nhân có thể là do "giới hạn độ phân giải".

Mạng lưới đường Pennsylvania: Mạng này phản ánh cấu trúc đường của Pennsylvania, trong đó các đỉnh đại diện cho giao lộ hoặc điểm cuối và các cạnh đại diện cho các đường nối các giao lộ hoặc điểm cuối này. Ở đây, chúng tôi đặt tham số $\lambda = 0,6$ cho Attractor và $i = 1,4$ cho MCL vì giá trị mặc định của hai thuật toán không thể mang lại kết quả tốt do mạng rất thưa thớt. Attractor cuối cùng đã xác định được 59.919 cụm có mô đun $= 0,856$ và ncut $= 25055$. MCL đạt được hiệu suất tương đương và tốt hơn thuật toán Metis. Louvain và Infomap chỉ tìm thấy một số lượng nhỏ các cộng đồng có quy mô bằng nhau (lần lượt là 492 và 208).

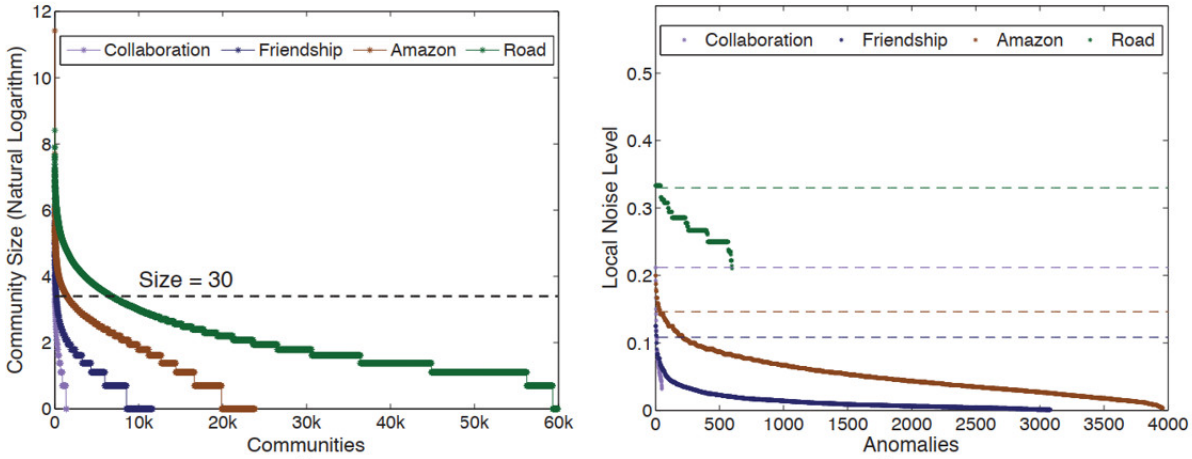
Tổng cộng, các thử nghiệm trên tất cả các mạng trong thế giới thực chứng minh rằng Attractor không chỉ cho phép phát hiện các cộng đồng có ý nghĩa trong các mạng có nhãn lớp (với hiệu suất cao nhất xét về tất cả các phép đo), mà còn mở rộng các mạng quy mô lớn và mang lại một biểu đồ tốt phân vùng theo các biện pháp bên trong (mô-đun và ncut) và bên ngoài (NMI, ARI và Độ tinh khiết) (xem Bảng 2, Bảng 3).

4.1.3. Phát hiện cộng đồng nhỏ và dị thường

Trong phần này, chúng tôi đánh giá liệu Attractor có cho phép xác định các cộng đồng nhỏ và các điểm bất thường có ý nghĩa hay không. Hình 16(a) vẽ biểu đồ phân bổ quy mô cộng đồng cho bốn mạng lớn trong thế giới thực và chúng ta có thể thấy rằng Attractor có thể tìm thấy nhiều cộng đồng nhỏ với quy mô khác nhau. Để chứng minh tiềm năng chất lượng cao của các cộng đồng, chúng tôi kiểm tra thêm chất lượng của các cộng đồng

nhỏ (kích thước < 30) trên mạng Amazon vì nó có cơ sở thực tế cho 5.000 cộng đồng hàng đầu. Điều thú vị cần lưu ý là 1458 cộng đồng nhỏ (kích thước 30) dẫn đến giá trị cao của NMI$= 0,941$, ARI $= 0,637$ và độ tinh khiết $= 0,989$, cho thấy đặc tính mong muốn của việc phát hiện cộng đồng nhỏ.

Ngoài ra, để kiểm tra xem các điểm bất thường được phát hiện có phải là các đỉnh nhiễu/bất thường tiềm ẩn hay không, chúng tôi đánh giá mức nhiễu địa phương của mỗi đỉnh, được định nghĩa là tỷ lệ của cấp độ đỉnh trên số lượng tất cả các liên kết của các đỉnh lân cận. Hình 16(b) mô tả mức nhiễu địa phương của tất cả các điểm bất thường phát sinh so với mức nhiễu trung bình (được biểu thị bằng các đường đứt nét) của Attractor trên bốn mạng trong thế giới thực, cung cấp bằng chứng tiềm năng cho việc phát hiện điểm bất thường hiệu quả.

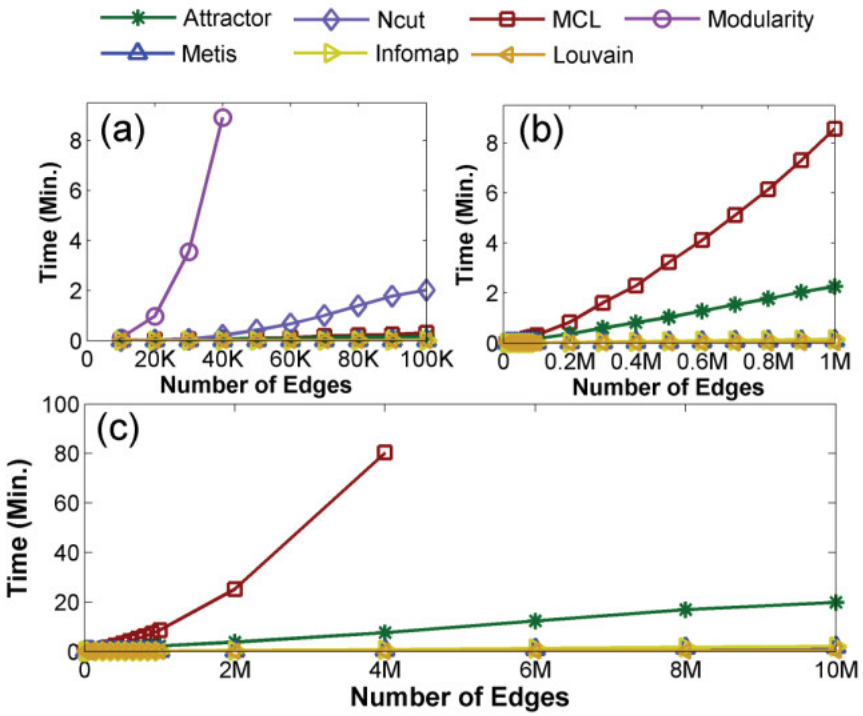


(a) Cộng đồng nhỏ (b) Bất thường

Hình 16: Đánh giá các cộng đồng nhỏ và sự bất thường.

4.1.4. Tốc độ chạy

Để đánh giá khả năng mở rộng của Attractor liên quan đến kích thước mạng, chúng tôi tạo ra một số mạng điểm chuẩn [32] với các kích thước cạnh khác nhau, từ mười nghìn đến mười triệu bằng cách cố định mức độ nút trung bình $k = 20$. Hình 17 cho thấy thời gian chạy cho các mức độ khác nhau. thuật toán phân cụm đồ thị. Chúng ta có thể quan sát thấy rằng Attractor nhanh hơn Modulity, Ncut và MCL vì độ phức tạp về thời gian của nó tuyến tính với $|E|$. Tuy nhiên, Attractor chậm hơn một chút so với các thuật toán phát hiện cộng đồng có thể mở rộng Metis, Louvain và Infomap.



Hình 17: Thời gian chạy của các thuật toán khác nhau.

4.2. So sánh thuật toán Biattractor với một số thuật toán khác

4.2.1. Thực nghiệm

Trong phần này, chúng tôi mong muốn đánh giá BiAttractor so với một số thuật toán nổi tiếng hiện có để phát hiện cộng đồng trong cả mạng hai phần tổng hợp. Trong phần này chúng tôi đã tham khảo kết quả so sánh trong tài liệu [17] đồng thời chạy thử nghiệm trên python (phụ lục A).

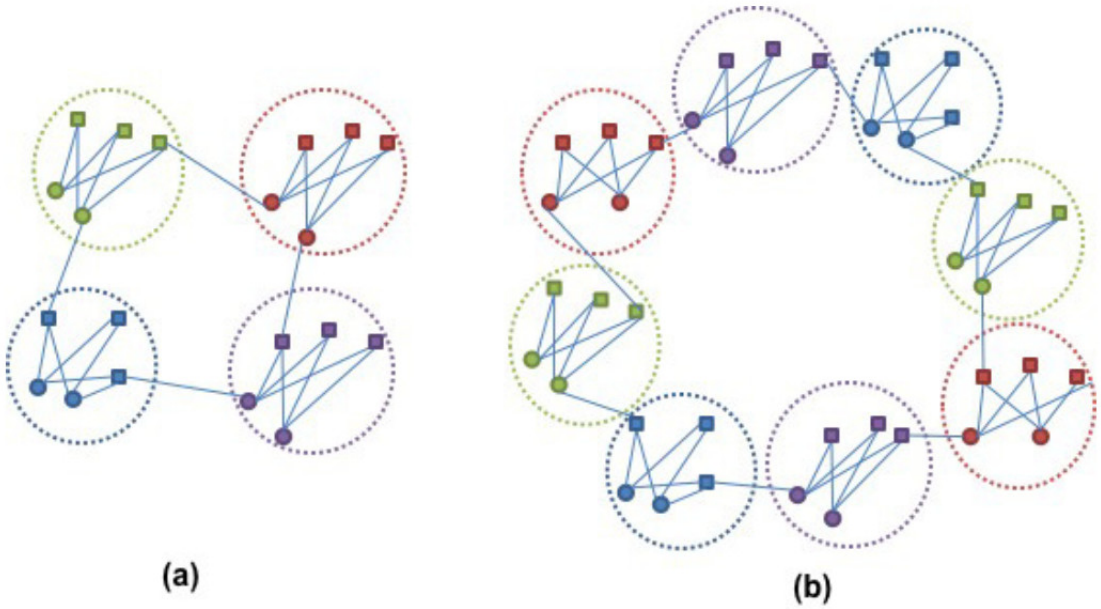
Adaptive BRIM Barber đã đề xuất BRIM [[21, 34]] dựa trên ý tưởng tối đa hóa lặp lại tính mô-đun Q_b trong mạng hai phần. Với mỗi lần lặp, Q_b được đảm bảo không giảm. Tuy nhiên, sự phân chia được xác định của các mạng hai phần dẫn đến mức tối đa địa phương chứ không phải mức tối đa toàn bộ của Q_b. Trong khi đó số lượng mô-đun cũng được xác định bằng cách tối đa hóa mô-đun Q_b.

LP BRIM Liu và cộng sự. đã mở rộng công việc của BRIM để đề xuất một phương pháp chung về truyền nhãn (LP) và BRIM có tên là LP BRIM [35]. Độ phức tạp về thời gian của nó tối đa là $O(n^2)$ (trong đó n là số đỉnh), có thể chấp nhận được để áp dụng trong các mạng thực.

AsymIntimacy Wang và cộng sự. các tham số bất đối xứng được xác định cho mức độ mật thiết giữa cùng loại đỉnh và loại đỉnh khác nhau [36]. Ban đầu, các đỉnh giống nhau

được hợp nhất thành các tập con do mức độ mật thiết không đối xứng. Sau đó, một loại đỉnh khác sẽ kết hợp vào các tập hợp con đó từ bước trước để tạo thành các cộng đồng cốt lõi. Mỗi cặp cộng đồng cốt lõi sẽ được hợp nhất nếu tỷ lệ giao nhau vượt quá ngưỡng. Quá trình tiếp tục cho đến khi không thể sáp nhập thêm cộng đồng cốt lõi nào nữa. Độ phức tạp về thời gian của nó là $O(2n^2 + mn)$ trong đó m là số cạnh và n là số đỉnh.

Biện pháp Để nghiên cứu và so sánh các phương pháp khác nhau, nhìn chung có hai loại biện pháp. Nếu trước đó phân tích cộng đồng trên các cơ sở mạng được đưa ra, Thông tin hỗn hợp đã được chuẩn hóa (NMI) [37] được áp dụng để chọn điểm từ $[0, 1]$. Mặt khác, modularity được sử dụng. Tính toán mô-đun ban đầu được chỉ định cụ thể cho các mạng đơn phương. Như vậy Barber đã mở rộng định nghĩa về cộng đồng thông qua mạng lưới bên trong là Q_b [21]. Giá trị cao hơn từ $[0, 1]$ biểu thị nhiều cạnh trong cộng đồng hơn so với mong đợi của null. Nhưng Q_b cũng có những chế độ giới hạn để giải quyết vấn đề giới hạn độ phân giải [38].



Hình 18: Vòng bicliques. (a) Một vòng gồm 4 hình vuông, mỗi hình tròn gồm hai hình tròn nối liền với nhau bằng ba hình vuông. (b) Một vòng gồm 8 bicliques có cùng quy tắc như ở (a).

4.2.2. Mạng tổng hợp

Một số phương pháp phụ thuộc vào việc tối ưu hóa tính mô đun. Tuy nhiên, các phương pháp như vậy có thể gặp vấn đề về giới hạn độ phân giải [38]. Chúng có những hạn chế trong việc phát hiện các cộng đồng nhỏ hơn một quy mô nhất định tùy thuộc vào tổng kích thước

của mạng và các kết nối bên trong của nó. Để chứng minh tính hiệu quả của BiAttractor, một vòng bicliques đã được thiết kế với các số bicliques khác nhau. Trên Hình 18 cho thấy một vòng được nối bằng 4 bicliques từ đầu đến đuôi và một vòng khác gồm 8 bicliques với quy tắc tương tự. Mỗi bicliques có hai đỉnh mode và hai hình tròn được kết nối hoàn toàn bằng ba hình vuông. Các đặc điểm cấu trúc tôpô cơ bản đã được tóm tắt trong Bảng 4. Các thí nghiệm tiếp theo được thực hiện trên các vòng bicliques có số lượng bicliques khác nhau.

Network	n_1	n_2	n	m	$\langle k \rangle$	C	r
4 *bicliques*	12	8	20	28	2.800	0.482	-0.5
8 *bicliques*	24	16	40	56	2.800	0.482	-0.5
16 *bicliques*	48	32	80	112	2.800	0.482	-0.5
64 *bicliques*	192	128	320	448	2.800	0.482	-0.5
128 *bicliques*	384	256	640	896	2.800	0.482	-0.5

Bảng 4: Đặc điểm cơ bản của các vòng bicliques.

Network	*BiAttractor*		*AdaptiveBRIM*		*LPBRIM*		*AsymIntimacy*	
	(NMI)	(N_C)	(NMI)	(N_C)	(NMI)	(N_C)	(NMI)	(N_C)
4 *bicliques*	**1.000**	**4**	**1.000**	4	**1.000**	4	0.714	4
8 *bicliques*	**1.000**	**8**	**1.000**	8	**1.000**	8	0.759	8
16 *bicliques*	**1.000**	**16**	0.934	15	0.802	13	0.785	16
64 *bicliques*	**1.000**	**64**	0.986	63	0.887	56	0.816	64
128 *bicliques*	**1.000**	**128**	0.993	127	0.900	113	0.826	128

Bảng 5: Hiệu suất của BRIM thích ứng, LP BRIM, BiAttractor và Bất đối xứng trên các vòng của bicliques. NMI cho biết độ chính xác của việc phát hiện cộng đồng hai chế độ. NC là số lượng cộng đồng được phát hiện bằng các phương pháp khác nhau.

Như đã biết từ Bảng 5, cả Adaptive BRIM và LP BRIM đều có thể phát hiện các cộng đồng hoàn hảo dưới dạng BiAttractor khi có 4 bicliques và 8 bicliques. Tuy nhiên, khi mạng là một vòng gồm 16 bicliques, LP BRIM thu được kết quả NMI($= 0,802$) với 13 cộng đồng. BRIM thích ứng có thể phát hiện 15 cộng đồng có NMI$= 0,934$. Bởi vì BiAttractor vẫn có thể tìm ra giải pháp hoàn hảo nên nó đạt được mức cải thiện độ chính xác là $7,07\%$ so với Adaptive BRIM và mức cải thiện độ chính xác lên tới $24,69\%$ so với LP BRIM. Nghiên cứu sâu hơn trên một vòng gồm 64 bicliques cũng cho kết quả tương tự. BiAttractor đạt được mức cải thiện độ chính xác $1,42\%$ so với Adaptive BRIM và cải thiện $12,74\%$ so với LP BRIM. Thí nghiệm cuối cùng của chúng tôi được thực hiện trong một vòng gồm 128

bicliques. BiAttractor vẫn cải thiện độ chính xác $0,70\%$ so với Adaptive BRIM và cải thiện $11,11\%$ so với LP BRIM. So sánh với AsymIntimacy, BiAttractor, Adaptive BRIM và LP BRIM luôn vượt trội hơn nó về mặt NMI trên tất cả các vòng của bicliques thể hiện trong Bảng 5. Nhưng AsymIntimacy thu được số lượng cộng đồng chính xác so với Adaptive BRIM và LP BRIM. Do đó, các phương pháp tối ưu hóa mô-đun như Adaptive BRIM, LP BRIM và phương pháp hợp nhất từ dưới lên AsymIntimacy có những vấn đề chung để phát hiện chính xác các vật thể nhỏ do giới hạn độ phân giải. Nhưng BiAttractor có thể phát hiện chính xác những bicliques nhỏ như vậy.

$Network$	n_1	n_2	n	m	$\langle k \rangle$	C	r
SW	18	14	32	89	5.563	0.328	-0.337
AR	136	5	141	160	2.270	0.781	-0.743
SCI	108	136	244	358	3.140	0.303	-0.171
CN	829	551	1380	1476	2.139	0.427	-0.166
MG	297	806	1103	2965	5.376	0.227	-0.300
PCD	680	739	1419	3690	1.746	0.407	-0.140
DW	89356	46215	135571	144342	2.129	0.447	-0.122
DP	48833	138839	187672	207268	2.209	0.514	-0.138

Bảng 6: Các đặc điểm cơ bản của mạng hai phần thực trong thí nghiệm này.

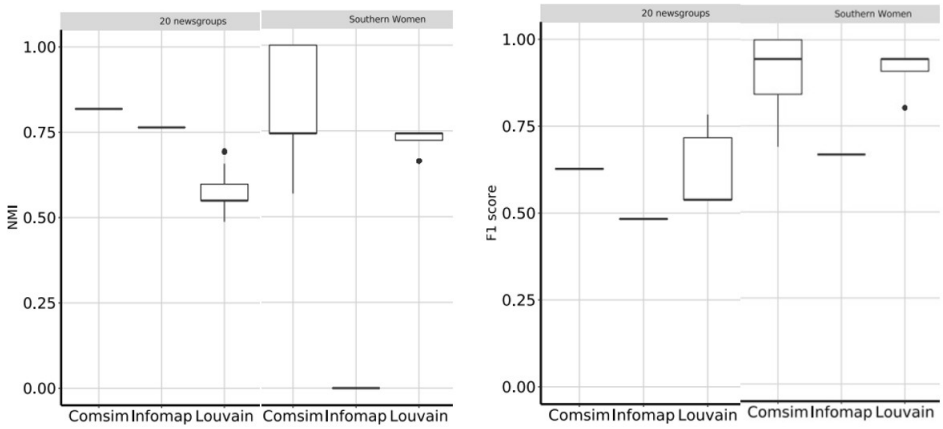
4.3. So sánh thuật toán ComSim với một số thuật toán khác

Trong phần này, chúng tôi mong muốn đánh giá ComSim so với một số thuật toán nổi tiếng hiện có để phát hiện cộng đồng trong cả mạng hai phần. Trong phần này chúng tôi đã tham khảo kết quả so sánh trong tài liệu [18] đồng thời chạy thử nghiệm trên python (phụ lục A).

Trước tiên, chúng tôi áp dụng thuật toán của mình cho hai mạng nhỏ nhưng được cung cấp khái niệm về các cộng đồng thực tế mà chúng tôi sử dụng làm tài liệu tham khảo để so sánh ba thuật toán.

Southern Women là một mạng lưới mô tả sự tham gia của 18 phụ nữ vào 14 sự kiện ở Hoa Kỳ được quan sát trong khoảng thời gian 9 tháng vào năm 1930. Mặc dù nhỏ nhưng bộ dữ liệu này rất thú vị vì nó đã được các nhà khoa học xã hội nghiên cứu rộng rãi để hiểu rõ hơn. Trong nghiên cứu này, chúng tôi sử dụng phân vùng được tìm thấy trong tài liệu làm cộng đồng sự thật cơ bản mà chúng tôi so sánh ba thuật toán.

20 newsgroups là kỷ lục của khoảng 50000 bài đăng được gửi bởi 30000 người dùng (bot) trên 20 nhóm thảo luận.



(a) NMI (b) F1-score

Hình 19: Đánh giá chất lượng phân vùng được thuật toán phát hiện trên hai mạng 20 newsgroups và Southern Women.

Hình 19 trình bày kết quả so sánh giữa ComSim và hai thuật toán cơ bản để phát hiện cộng đồng hai bên. Tất cả các thuật toán được áp dụng trên tập dữ liệu 20 newsgroups và Southern Women. Các ô hình hộp cung cấp các giá trị tối đa, tối thiểu và trung bình.

Vì chúng tôi có phân vùng thực tế cơ bản cho tập dữ liệu nên trước tiên chúng tôi sử dụng Thông tin tương hỗ được chuẩn hóa thông thường (ví dụ: NMI) để tính toán khoảng cách giữa các phân vùng được phát hiện và phân vùng thực tế cơ bản. Hình 19a cho thấy rằng đối với cả hai bộ dữ liệu, ComSim là thuật toán đề xuất phân vùng trung bình tốt nhất. Người ta cũng có thể nhận thấy rằng Infomap tạo phân vùng tốt cho mạng 20 newsgroups và Louvain phân vùng tốt cho mạng Southern Women.

Để cung cấp quan điểm thứ hai, chúng tôi cũng sử dụng F1-score, một thước đo cổ điển để đánh giá hiệu suất của các thuật toán dự đoán (xem [39] để biết ví dụ về F1-score được sử dụng trong bối cảnh các vấn đề phát hiện cộng đồng). Hình 19b một lần nữa cho thấy ComSim là thuật toán phát hiện cộng đồng tốt nhất cho cả hai tập dữ liệu. Điều thú vị là đối với số liệu này, Louvain dường như đề xuất các phân vùng trung bình tốt cho cả hai tập dữ liệu.

Nhìn chung, có vẻ như ComSim đề xuất các cộng đồng rõ ràng khi so sánh với các phân vùng thực tế của mạng lưới hai bên. Phần tiếp theo dự định nghiên cứu hoạt động của thuật toán trên mạng quy mô lớn.

	Southern women	20 newsgroups	IMDb
|U|/|v|/số cạnh	18/14/89	20/30K/42K	122K/118K/531K
COMSIM	1,7 ms / 11,5 MB	1,1 s / 30 MB	33,5 s / 591,6 MB
Infomap	13 ms / 10,7 MB	951 ms / 6.3 MB	100s / 374 MB
Louvain	11 ms / 6,5 MB	86 ms / 10,1 MB	21 s / 43 MB

Bảng 7 Hiệu suất về thời gian thực hiện và mức bộ nhớ tối đa cho bốn thuật toán.

Bảng 7 trình bày hiệu suất về thời gian thực hiện và mức bộ nhớ tối đa cho ba thuật toán trên ba bộ dữ liệu. Điều này cho thấy Louvain vẫn là thuật toán hiệu quả nhất về cả thời gian và bộ nhớ, cho thấy chỉ chậm hơn trên tập dữ liệu nhỏ nhất.

Tuy nhiên, cần nhấn mạnh ở đây việc sử lý dữ liệu của Louvain trong Bảng 7 đã được ghi lại sau phép chiếu U Điều này có nghĩa là một phần tải tính toán liên quan đến hàm θ đã được tránh, điều này không xảy ra với các thuật toán khác. Do đó, các thuật toán chạy sẽ có lợi thế hơn khi tiếp cận Louvain.

Về vấn đề đó, điều đáng chú ý là thuật toán của chúng tôi mang lại kết quả tốt. Đặc biệt trên IMDb, ComSim chỉ chậm hơn Louvain một chút và nhanh hơn Infomap ba lần.

KẾT LUẬN

Trong phần này chúng tôi sẽ tóm tắt nội dung chính của luận văn.

Luận văn này trình bày về bài toán phát hiện cộng đồng mạng:

Chương 1 trình bày một số kiến thức nền liên quan đến bài toán phát hiện cộng đồng trên mạng hay phân cụm đồ thị. Trong chương này chúng tôi đã trình bày lại một số định nghĩa về lý thuyết đồ thị có liên quan đến luận văn. Ngoài ra chúng tôi còn trình bày một số lý thuyết liên quan đến bài toán phát hiện cộng đồng mạng và một số ví dụ điển hình mà tôi đã sưu tầm được trong sách tham khảo [5], các tài liệu liên quan khác [4] và một số ví dụ chúng tôi nhận thấy trong thế giới thực.

Chương 2 trình bày chi tiết về khoảng cách Jaccard (một loại khoảng cách trực tiếp giữa hai điểm), ảnh hưởng từ các đỉnh lân cận chung và riêng tới khoảng cách giữa các điểm. Sau đó trình bày thuật toán Attractor phát hiện cộng đồng mạng dựa trên khoảng cách Jaccard. Trong chương này tôi đã lựa chọn bài báo [13] của Junming Shao, Zhichao Han, Qinli Yang, TaoZhou làm tài liệu tham khảo chính. Tôi đã dịch bài báo và trình bày, diễn giải lại bài báo này.

Chương 3 trình bày một số lí thuyết liên quan đến đồ thị hai phần sau đó trình bày thuật toán BiAttractor và ComSim để phát hiện cộng đồng trên mạng hai phần. Trong mục 3.1 chúng tôi trình bày về thuật toán BiAttactor. Chúng tôi sử dụng bài báo [17] của Hong-liang Sun, Eugene Ch'ng, Xi Yong, Jonathan M. Garibaldi, Simon See và Duan-bing Chen làm tài liệu tham khảo chính. Tôi đã lựa chọn tài liệu này vì trong tài liệu này chính là phần phát triển và mở rộng của bài báo [13] và nó đưa ra khoảng cách Jaccard cho đồ thị hai phần và thuật toán BiAttractor một thuật toán phát hiện cộng đồng cho đồ thị hai phần điều mà thuật toán Attractor chưa thực hiện được. Trong mục 3.2 chúng tôi trình bày về thuật toán ComSim. Chúng tôi sử dụng bài báo [18] của Raphael Tackx, Fabien Tarissan và Jean-Loup Guillaume làm tài liệu tham khảo chính. Lý do để tôi chọn thuật toán này trình bày vào luận văn của mình vì thuật toán này cho ta một cách tiếp cận trực quan. Hơn thế nữa, trong thực tế có rất nhiều ví dụ về các đồ thị hai phần mà người ta chỉ quan tâm đến việc phân cụm một phần của nó. Thuật toán này làm rất tốt việc phân cụm một phần của đồ thị hai phần. Tuy nhiên chúng ta cũng có những bài toán cần phân cụm cả hai phần của đồ thị thì thuật toán này chưa giải quyết được. Chúng tôi đã mở rộng và đề xuất một cách

phân cụm cho phần còn lại của đồ thị ở mục 3.2.3.

Trong chương 4, chúng tôi đã so sánh các thuật toán chúng tôi trình bày ở trên với các thuật toán phổ biến khác. Trong chương này chúng tôi vẫn sử dụng 3 bài báo [13, 17, 18] làm tài liệu tham khảo chính của mình. Ngoài ra chúng tôi đã lập trình lại các thuật toán bằng ngôn ngữ Python (xem chi tiết ở phụ lục A) để chạy một số thí nghiệm trên đó.

Chương trình chạy thuật toán Attractor, BiAttractor và ComSim đầy đủ được trình bày trong phụ lục của luận văn.

TÀI LIỆU THAM KHẢO

[1] Raphael Michele Coscia, *The Atlas for the Aspiring Network Scientist*. 2021. arXiv:2101.00863v2.

[2] Katharina A. Zweig, *Network Analysis Literacy*. Springer Vienna, 2016.

[3] Ulrik Brandes, Garry Robins, Ann McCranie, Stanley Wasserman, *What is network science?* Network Science, 1(1), trang 1–15, 2013.

[4] Nguyễn Xuân Dũng, *Nghiên cứu các thuật toán rút gọn đồ thị và ứng dụng để phát triển cộng đồng trên mạng xã hội*. Luận văn tiến sĩ, 2021.

[5] Xin Huang, *Community Search over Big Graphs*. Morgan Claypool publishers, 2019.

[6] Martin Rosvall, Jean-Charles Delvenne, Michael T. Schaub, Renaud Lam-biotte, *Different approaches to community detection. In Patrick Doreian, Vladimir Batagelj, and Anuska Ferligoj, editors*. Advances in Network Clustering and Block-modeling, trang 105–119, 2020.

[7] Michele Coscia, Fosca Giannotti, Dino Pedreschi, *A classification for community discovery methods in complex networks*. Statistical Analysis and Data Mining: The ASA Data Science Journal, 4(5), trang 512–546, 2011.

[8] Santo Fortunato, *Community detection in graphs*. Physics Reports, 486(3-5), trang 75–174, 2010.

[9] Vinh Loc Dao, Cecile Bothorel, Philippe Lenca, *Community structure: Acomparative evaluation of community detection methods*. Network Science, 8(1), trang 1–41, 2020.

[10] Zhao Yang, René Algesheimer và Claudio J. Tessone, *A comparative analysis of community detection algorithms on artificial networks*. Scientific Reports, 6(1), trang 1–18, 2016.

[11] Amir Ghasemian, Homa Hosseinmardi, Aaron Clauset, *Evaluating overfit and underfit in models of network community structure.*. IEEE Transactions on Knowledge and Data Engineering, 32(9), trang 1722–1735, 2019.

[12] Liudmila Prokhorenkova, Alexey Tikhonov, *Community detection through likelihood optimization: in search of a sound model*. In The World Wide Web Conference, trang 1498–1508, 2019.

[13] Junming Shao, Zhichao Han, Qinli Yang và Tao Zhou, *Community Detection based on Distance Dynamics*. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining August 2015, trang 1075 -1084, 2015.

[14] Christian Hennig và Bernhard Hausdorf. *Design of dissimilarity measures: A new dissimilarity between species distribution areas*. In Data Science and Classification, trang 29–37. Springer, 2006.

[15] Nir Ailon, Yudong Chen, Huan Xu, *Breaking the small cluster barrier of graph clustering*. In ICML, trang 995–1003, 2013.

[16] Santo Fortunato, Marc Barthélemy, *Resolution limit in community detection*. PNAS, 104(1), trang 36–41, 2007.

[17] Hong-liang Sun, Eugene Ch'ng, Xi Yong, Jonathan M. Garibaldi, Simon See và Duanbing Chen, *A Fast Community Detection Method in Bipartite Networks by Distance Dynamics*. Physica A: Statistical Mechanics and its Applications, trang 108–1120, 2018.

[18] Raphael Tackx, Fabien Tarissan và Jean-Loup Guillaume, *ComSim : A bipartite community detection algorithm using cycle and node's similarity*. Complex Networks 2017: Complex Networks and Their Applications VI, trang 278–289, 2017.

[19] T. Zhou, L. Y. Lu Y.và C. Zhang, *Predicting missing links via local information*. European Physical Journal B 71 (4), trang 623–630, 2009.

[20] L. Y. Lu, C. H. Jin và T. Zhou, *Similarity index based on local paths for link prediction of complex networks*. Physical Review E 80 (4), 2009.

[21] M. J. Barber, *Modularity and community detection in bipartite networks*. Physical Review E 76 (6), 2007.

[22] Paul Jaccard, *Le coefficient generique et le coefficient de communaute dans la flore marocaine*. Impr. Commerciale, 1926.

[23] Lada A Adamic và Eytan Adar, *Friends and neighbors on the web*. Social networks, 25(3), trang 211–230, 2003.

[24] Jianbo Shi, Jitendra Malik, *Normalized cuts and image segmentation*. EEE TPAMI, 22(8), trang 888–905, 2000.

[25] Mark EJ Newman, *Modularity and community structure in networks*. Proceedings of the National Academy of Sciences, 103(23), trang 8577–8582, 2006.

[26] George Karypis, Vipin Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*. SIAM Journal on scientific Computing, 20(1), trang 359–392, 1998.

[27] Stijn Van Dongen, *A cluster algorithm for graphs*. Report-Information systems,(10), trang 1–40, 2000.

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

[29] Martin Rosvall, Carl T Bergstrom, *Maps of random walks on complex networks reveal community structure*. PNAS, 105(4), trang 1118–1123, 2008.

[30] Alexander Strehl, Joydeep Ghosh, *Cluster ensembles—a knowledge reuse framework for combining multiple partitions*. The Journal of Machine Learning Research, 3, trang 583–617, 2003.

[31] William M Rand, *Objective criteria for the evaluation of clustering methods*. Journal of the American Statistical association, 66(336), trang 846–850, 1971.

[32] Andrea Lancichinetti, Santo Fortunato, Filippo Radicchi, *Benchmark graphs for testing community detection algorithms*. Physical review E, 78(4), 2008.

[33] Jaewon Yang and Jure Leskovec, *Defining and evaluating network communities based on ground-truth*. In IEEE ICDM, trang 745–754, 2012.

[34] M. J. Barber, M. Faria, L. Streit, Strogan, *Searching for communities in bipartite networks*. In Stochastic and Quantum Dynamics of Biomolecular Systems 1021, trang 171-182, 2008.

[35] X. Liu, T. Murata, *Community detection in large-scale bipartite networks*. Transactions of the Japanese Society for Artificial Intelligence 25 (1), trang 16-24, 2010.

[36] X. Wang, X. Qin, *Asymmetric intimacy and algorithm for detecting communities in bipartite networks*. Physica A: Statistical Mechanics and its Applications 462 trang 8577-8582, 2007.

[37] P. Zhang, *Evaluating accuracy of community detection using the relative normalized mutual information*. Journal of Statistical Mechanics: Theory and Experiment 11, 2015.

[38] S. Fortunato, M. Barthelemy, *Resolution limit in community detection*. Proceedings of the National Academy of Sciences USA 104 (1), trang 36-41, 2007.

[39] Jaewon Yang và Jure Leskovec, *Overlapping community detection at scale: a nonnegative matrix factorization approach*. In Proceedings of the sixth ACM international conference on Web search and data mining, trang 587–596, 2013.

[40] Santo Fortunato, Darko Hric, *Community detection in networks: A user guide*. Physics Reports, 659, trang 1–44, 2016.

[41] Satu Elisa Schaeffer, *Graph clustering*. Computer Science Review, 1(1), trang 27–64, 2007.

[42] W. Chen, Z. M. Liu, X. R. Sun, Y. J. Wang, *A game-theoretic framework to identify overlapping communities in social networks*. Data Mining and Knowledge Discovery 21, trang 224-240, 2010.

[43] Tao Zhou, Linyuan Lu và Yi-Cheng Zhang, *The European Physical Journal B-Condensed Matter and Complex Systems*. 71(4), trang 623–630, 2009.

Phụ lục A

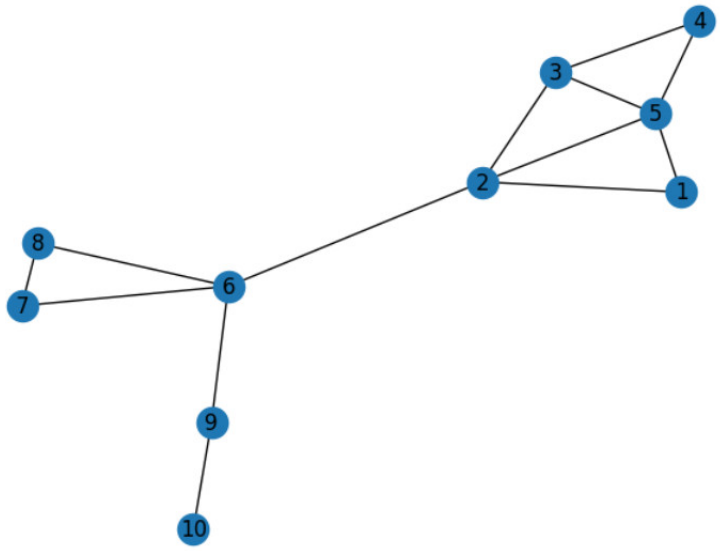
THUẬT TOÁN ATTRACTOR, BIATTRACTOR VÀ COMSIM TRÊN PYTHON

Thuật toán Attractor

```
[1] import networkx as nx
    import numpy
    import math
```

```
G = nx.from_edgelist([(1,2), (1,5), (2, 5), (2, 3), (2, 6), (3, 5),
                      (3, 4), (4, 5), (6, 7), (6, 8), (6, 9), (7, 8),(9,10)])
```

```
nx.draw(G, with_labels=True)
```



✓ 0 giây hoàn thành lúc 17:54

```
[4]  # Fomular 3: Compute Direct Distance (Jaccard Distance)
     def direct_distance(u,v, G):
       # d = 1-( {số đỉnh kề chung}/{số đỉnh kề của hai đỉnh})
       adj_u = set()
       adj_v = set()
       for node in G.nodes():
         if G.has_edge(u, node):
           adj_u.add(node)
         if G.has_edge(v, node):
           adj_v.add(node)
       d = 1 - len(adj_u.intersection(adj_v))/len(adj_u.union(adj_v))
       return d
```

```
[5]  def DI(u, v, G:nx.Graph(), f=math.sin):
       global distance
       return -(f(1-distance[u][v])/G.degree(u)+f(1-distance[u][v])/G.degree(v))
```

```
[6]  def CN(u,v, G:nx.Graph()):
       adj_u = set()
       adj_v = set()
       for node in G.nodes():
         if G.has_edge(u, node):
           adj_u.add(node)
         if G.has_edge(v, node):
           adj_v.add(node)
       return adj_u.intersection(adj_v)
```

```
[7]  def EN(u,v, G:nx.Graph()):
       adj_u = set()
       adj_v = set()
       for node in G.nodes():
         if G.has_edge(u, node):
           adj_u.add(node)
         if G.has_edge(v, node):
           adj_v.add(node)
       adj = adj_u.intersection(adj_v)
       return adj_u-adj
```

```
[8]  def CI(u,v,G, f=math.sin):
       global distance
       cn = CN(u,v, G)
       result = 0
       for x in cn:
         temp = 1/G.degree(u)*f(1-distance[x][u])*(1-distance[x][u])+\
          1/G.degree(v)*f(1-distance[x][v])*(1-distance[x][v])
         result += -temp
       return result
```

```
[9]  def rho(x, v, G, l=0.5):
       global distance
       tmp = 1 - distance[x][v]
       if tmp >= l:
         return tmp
       else:
         return tmp-l
```

```
[10] def EI(u, v, G, f, l):
       global distance
       adj_u = EN(u, v, G)
       adj_v = EN(v, u, G)

       sigma_u = 0
       for x in adj_u:
         sigma_u += 1/G.degree(u)*f(1-distance[x][u])*rho(x,u,G,l)
       sigma_v = 0
       for x in adj_v:
         sigma_v += 1/G.degree(v)*f(1-distance[x][v])*rho(x,v,G,l)
       return -(sigma_u+sigma_v)
```

```
[11] def update_variable(f, l):
       global di_dictionary
       global ci_dictionary
       global ei_dictionary
       di_dictionary = {}
       ci_dictionary = {}
       ei_dictionary = {}
       for u in G.nodes():
         di_dictionary[u] = {}
         ci_dictionary[u] = {}
         ei_dictionary[u] = {}

         for v in G.nodes():
           di_dictionary[u][v] = DI(u,v,G, f)
           ci_dictionary[u][v] = CI(u,v,G, f)
           ei_dictionary[u][v] = EI(u, v, G, f, l)
       return di_dictionary, ci_dictionary, ei_dictionary
```

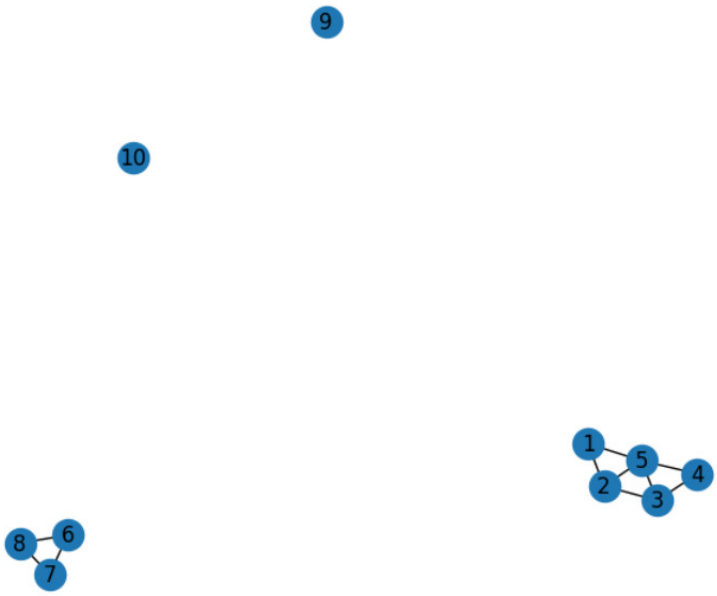
```
[12] def Attactor_algorithm(G):
       alone_node = []
       for node in G:
         if G.degree(node)==0:
           alone_node.append(node)
       G.remove_nodes_from(alone_node)

       global distance
       global di_dictionary
       global ci_dictionary
       global ei_dictionary

       f = math.sin
       l = 0.5
       for u in G.nodes():
         distance[u] = {}
         for v in G.nodes():
           distance[u][v] = direct_distance(u,v,G)

       flag = True
       while flag:
         flag=False
         di_dictionary, ci_dictionary, ei_dictionary = update_variable(f, l)
         for u,v in G.edges():
           #de^t: khoảng cách ban đầu
           if distance[u][v] > 0 and distance[u][v] < 1:
             delta_det = di_dictionary[u][v]+ ci_dictionary[u][v]+ ei_dictionary[u][v]
             if delta_det!=0:
               distance[u][v]+=delta_det
               if distance[u][v]>1 :
                 distance[u][v] = 1
               elif distance[u][v]<0:
                 distance[u][v] = 0
               flag=True
       for u,v in G.edges():
         if distance[u][v] == 1:
           G.remove_edge(u,v)
       return G
```

```
[13] distance = {}
     di_dictionary = {}
     ci_dictionary = {}
     ei_dictionary = {}
     G = Attactor_algorithm(G)
```

```
[14] from networkx.generators import ego
     nx.edges(G)
```

```
EdgeView([(1, 2), (1, 5), (2, 5), (2, 3), (5, 3), (5, 4), (3, 4), (6, 7), (6, 8), (7, 8)])
```

```
[15] nx.draw(G, with_labels=True)
```



Thuật toán Biattractor

```
[1] import networkx as nx
```

```
[2] def find_N(u, G: nx.Graph)->set:
      """ Neighborhood of vertex u consists of adjacent vertices of u

      Input:
        - u: A node of Graph G
        - G (nx.Graph):  Undirected and unweight bipartite network
      Output:
        - res (set): Set node
      """
      res = set()
      for v in list(G.nodes()):
        if G.has_edge(u,v):
          res.add(v)
      return res
```

```
[3] def compute_d_jac(u, v, G: nx.Graph)->float:
      """Compute Jaccard Distance between u and v (Eq. 5)
      N(u) \cap N(v) is null set if u is connected with v because N(u) and N(v) are different types of vertices

      Input:
        - u: Node
        - v: Node
        - G (nx.Graph):  Undirected and unweight bipartite network
      Output:
        - d (float): Jaccard Distance
      """
      # d = 1-( {số đỉnh kề chung}/{số đỉnh kề của hai đỉnh})\
      # N(u) \cap N(v) is null set if u is connected with v because N(u) and N(v) are different types of vertices
      if G.has_edge(u, v):
        return 1
      N_u = find_N(u, G)
      N_v = find_N(v, G)
      d = 1 - len(N_u.intersection(N_v))/len(N_u.union(N_v))
      return d
```

```
[4] def compute_d_local_jac(u, v, G: nx.Graph)-> float:
      """Compute Local Jaccard Distance between u and v (Eq. 6)

      Input:
        - u: Node
        - v: Node
        - G (nx.Graph):  Undirected and unweight bipartite network
      Output:
        - d (float): Local Jaccard Distance
      """
      EN_u = find_N(u, G) - {v}
      EN_v = find_N(v, G) - {u}
      # print('---*---')
      # print(f'EN_{u}:', EN_u)
      # print(f'EN_{v}:', EN_v)
      if len(EN_u)*len(EN_v)==0:
        return  1
      d_1 = sum([compute_d_jac(u, _v, G) for _v in EN_v])
      d_2 = sum([compute_d_jac(v, _u, G) for _u in EN_u])
      d = 1/2*(1/len(EN_v)*d_1+1/len(EN_u)*d_2)
      # print(f'Local Jaccard Distance between {u}, {v}: {d}')
      return d
```

[5]
```python
import math
from typing import Callable

def compute_direct_influence(u, v, G: nx.Graph, f: Callable=math.sin):
  """Compute Direct Influence between u and v
  We change the distance function d(u, v)
  from Jaccard Distance to Local Jaccard Distance
  because Jaccard Distance always fails to denote node similarity in bipartite networks.
  Input:
    - u: Node
    - v: Node
    - G (nx.Graph):  Undirected and unweight bipartite network
    - f (function): sin(x)/ cos(X),.... mapping x -> [0,1]

  Output:
    - di (float): Direct Influence between u and v
  """
  global local_jaccard_distance
  d_uv = local_jaccard_distance[u][v] #compute_d_jac(u, v, G)
  direct_influ = -(f(1-d_uv)/G.degree(u)+f(1-d_uv)/G.degree(v))
  return direct_influ
```

[6]
```python
def rho(x, v, threshold=0.5):
  global local_jaccard_distance
  tmp = 1 - local_jaccard_distance[x][v]
  if tmp >= threshold:
    return tmp
  else:
    return tmp - threshold
```

[7]
```python
def compute_exclusive_influence(u, v, G: nx.Graph, f: Callable=math.sin):
  """"""
  global local_jaccard_distance
  EN_u = find_N(u, G) - {v}
  EN_v = find_N(v, G) - {u}

  threshold = 0.5
  sigma_u = 0
  for x in EN_u:
    sigma_u += 1/G.degree(u)*f(1-local_jaccard_distance[x][u])*rho(x,u,threshold)
  sigma_v = 0
  for x in EN_v:
    sigma_v += 1/G.degree(v)*f(1-local_jaccard_distance[x][v])*rho(x,v,threshold)
  return -(sigma_u+sigma_v)
```

```
[8]  def update_variable(G):
       global local_jaccard_distance
       global direct_influence
       global exclusive_influence

       f = math.sin
       # Update global variable with new graph G
       # for u, v in itertools.product(set(G), set(G)):
       for u in set(G):
         local_jaccard_distance[u] = {}
         for v in set(G):
           local_jaccard_distance[u][v] = compute_d_local_jac(u, v, G)

       for u in set(G):
         # local_jaccard_distance[u] = {}
         direct_influence[u] = {}
         exclusive_influence[u] = {}
         for v in set(G):
           # local_jaccard_distance[u][v] = compute_d_local_jac(u, v, G)
           direct_influence[u][v] = compute_direct_influence(u, v, G, f)
           exclusive_influence[u][v] = compute_exclusive_influence(u, v, G, f)
       return local_jaccard_distance, direct_influence, exclusive_influence
```

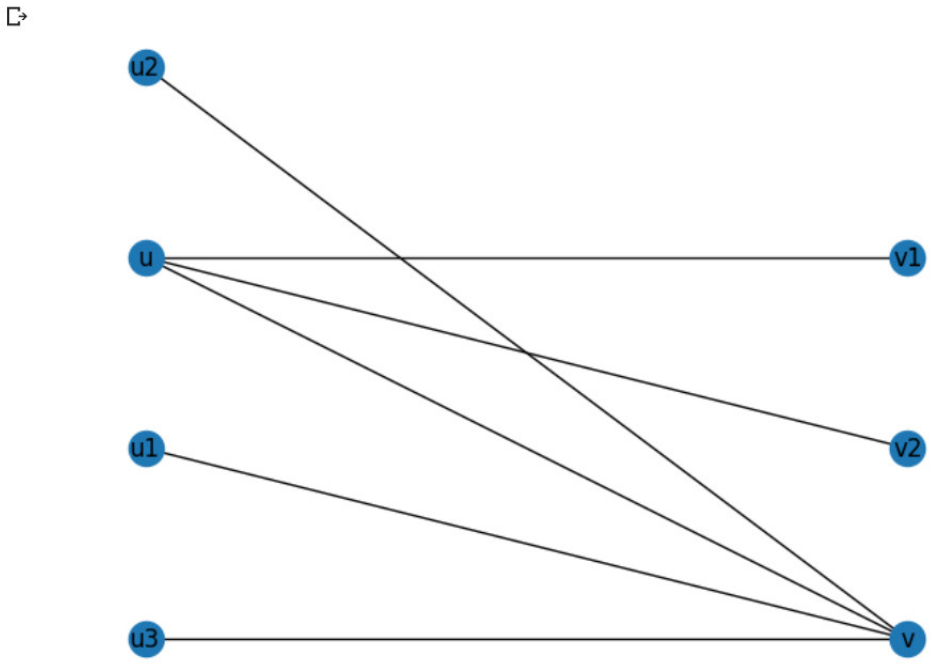
```
[9]  def biattractor_algorithm(G):
       global local_jaccard_distance
       global direct_influence
       global exclusive_influence

       local_jaccard_distance, direct_influence, exclusive_influence = update_variable(G)

       flag = True # Not convergnece from all edges
       iter = 0
       while flag:
         print(f"Iter {iter}: {local_jaccard_distance}")
         iter+=1
         flag = False
         for u,v in G.edges():
           if local_jaccard_distance[u][v] > 0 and local_jaccard_distance[u][v] < 1:
             tmp = local_jaccard_distance[u][v]
             local_jaccard_distance[u][v] = local_jaccard_distance[u][v] + direct_influence[u][v] + exclusive_influence[u][v]
             if tmp != local_jaccard_distance[u][v]:
               print(f'Update Distance between Node {u} and {v}: {tmp} --> {local_jaccard_distance[u][v]}')
               flag = True
           elif local_jaccard_distance[u][v] <= 0:
             local_jaccard_distance[u][v] = 0
           elif local_jaccard_distance[u][v] >= 1:
             local_jaccard_distance[u][v] = 1
       for u,v in G.edges():
         if local_jaccard_distance[u][v] == 1:
           G.remove_edge(u,v)
       return G
```

```
[10] # A Toy Sample Fig. 2 (a)
     B = nx.Graph()
     # Add nodes with the node attribute "bipartite"
     B.add_nodes_from(['u', 'u1', 'u2', 'u3'], bipartite=0)
     B.add_nodes_from(["v", "v1", "v2"], bipartite=1)
     # Add edges only between nodes of opposite node sets
     B.add_edges_from([('v1', 'u'), ('v2', 'u'), ('u', 'v'), ('v', 'u1'), ('v', 'u2'), ('v', 'u3')])
```

```
[11] # nx.draw(B, with_labels=True)
```

```
[12] from networkx.algorithms import bipartite
     import matplotlib.pyplot as plt
```

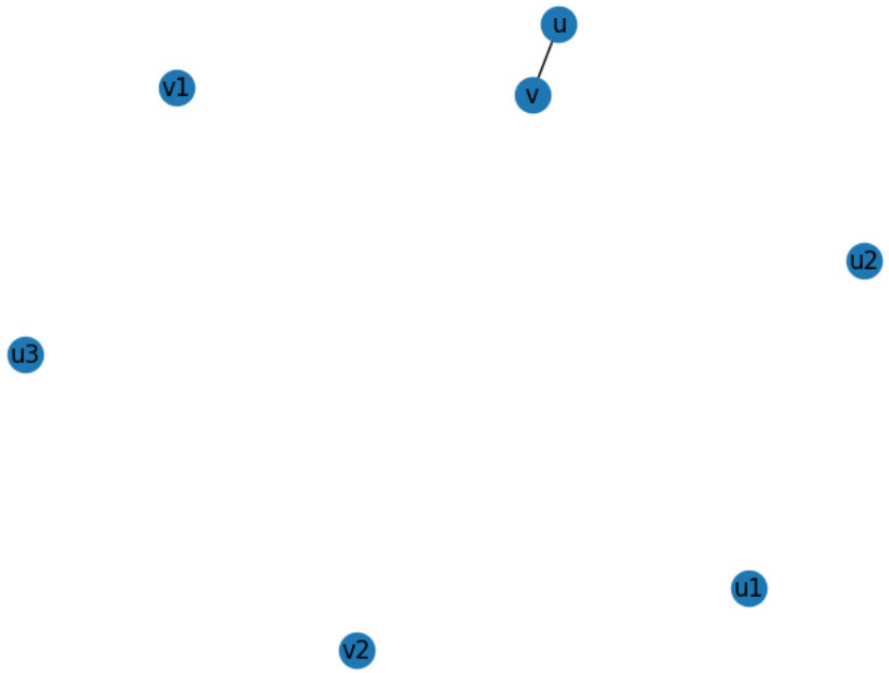
```
[13] X, Y = bipartite.sets(B)
     pos = dict()
     pos.update( (n, (1, i)) for i, n in enumerate(X) ) # put nodes from X at x=1
     pos.update( (n, (2, i)) for i, n in enumerate(Y) ) # put nodes from Y at x=2
     nx.draw(B, pos=pos, with_labels=True)
     plt.show()
```



```
[14] local_jaccard_distance = {}
     direct_influence = {}
     exclusive_influence = {}
     G = biattractor_algorithm(B)
```

```
Iter 0: {'u3': {'u3': 1.0, 'v': 1, 'u1': 1.0, 'u': 1.0, 'v2': 0.7083333333333334, 'v1': 0.7083333333333334, 'u2'
Update Distance between Node u and v: 0.7083333333333333 --> 0.5405964748042751
Iter 1: {'u3': {'u3': 1.0, 'v': 1, 'u1': 1.0, 'u': 1.0, 'v2': 0.7083333333333334, 'v1': 0.7083333333333334, 'u2'
Update Distance between Node u and v: 0.5405964748042751 --> 0.37285961627521697
Iter 2: {'u3': {'u3': 1.0, 'v': 1, 'u1': 1.0, 'u': 1.0, 'v2': 0.7083333333333334, 'v1': 0.7083333333333334, 'u2'
Update Distance between Node u and v: 0.37285961627521697 --> 0.20512275774615885
Iter 3: {'u3': {'u3': 1.0, 'v': 1, 'u1': 1.0, 'u': 1.0, 'v2': 0.7083333333333334, 'v1': 0.7083333333333334, 'u2'
Update Distance between Node u and v: 0.20512275774615885 --> 0.037385899217100726
Iter 4: {'u3': {'u3': 1.0, 'v': 1, 'u1': 1.0, 'u': 1.0, 'v2': 0.7083333333333334, 'v1': 0.7083333333333334, 'u2'
Update Distance between Node u and v: 0.037385899217100726 --> -0.1303509593119574
Iter 5: {'u3': {'u3': 1.0, 'v': 1, 'u1': 1.0, 'u': 1.0, 'v2': 0.7083333333333334, 'v1': 0.7083333333333334, 'u2'
```

```
[15] nx.draw(B, with_labels=True)
```



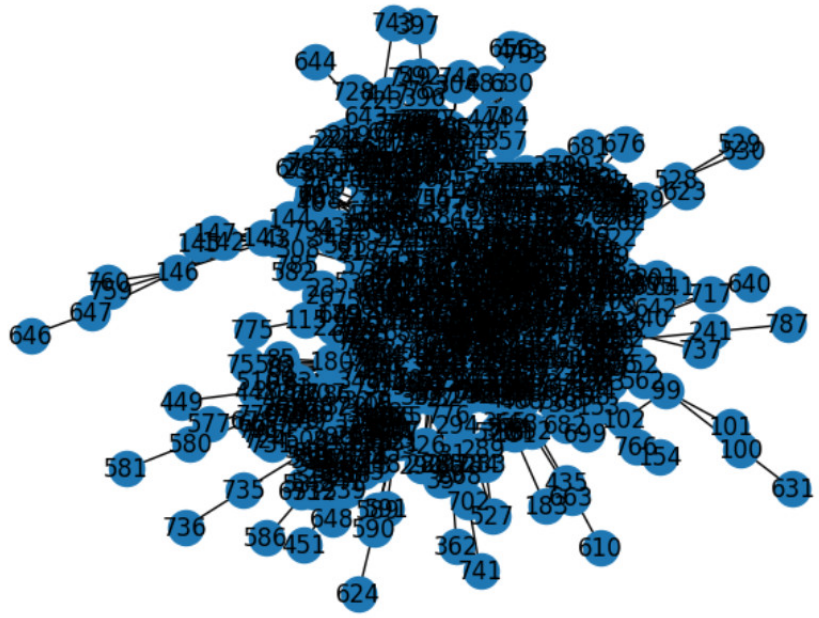
```
[22] !unzip '/content/data_undirected2 (1).zip'
```

```
Archive:  /content/data_undirected2 (1).zip
   creating: data_undirected2 (1)/
  inflating: data_undirected2 (1)/out.arenas-pgp_1.txt
  inflating: data_undirected2 (1)/out.as20000102_1.txt
  inflating: data_undirected2 (1)/out.asoiaf_1.txt
  inflating: data_undirected2 (1)/out.dbpedia-similar_1.txt
  inflating: data_undirected2 (1)/out.dnc-corecipient_1.txt
  inflating: data_undirected2 (1)/out.maayan-vidal_1.txt
  inflating: data_undirected2 (1)/out.moreno_names_names_1.txt
  inflating: data_undirected2 (1)/out.moreno_propro_propro_1.txt
  inflating: data_undirected2 (1)/out.petster-friendships-hamster-uniq_1.txt
  inflating: data_undirected2 (1)/out.petster-hamster-household_1.txt
```

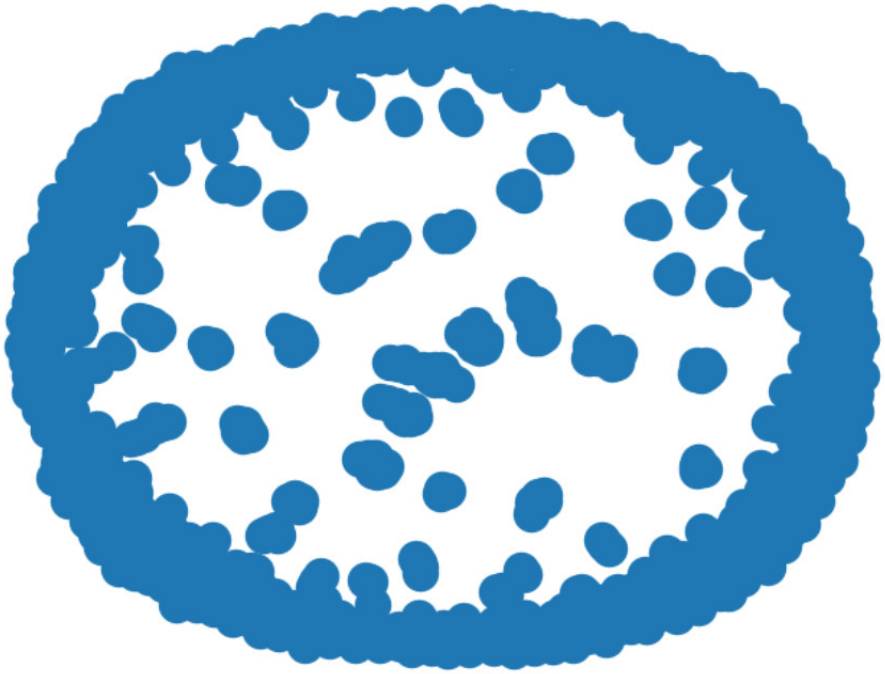
```
[23] link_data = '/content/data_undirected2 (1)/out.asoiaf_1.txt'
     g = nx.read_adjlist(link_data, create_using=nx.Graph, nodetype = int)
     mapping = {list(g.nodes())[i]:i for i in range(len(list(g.nodes())))}
     g = nx.relabel_nodes(g, mapping)
     print("G = (",g.number_of_nodes(),",",g.number_of_edges(),")")
```

```
G = ( 796 , 2823 )
```

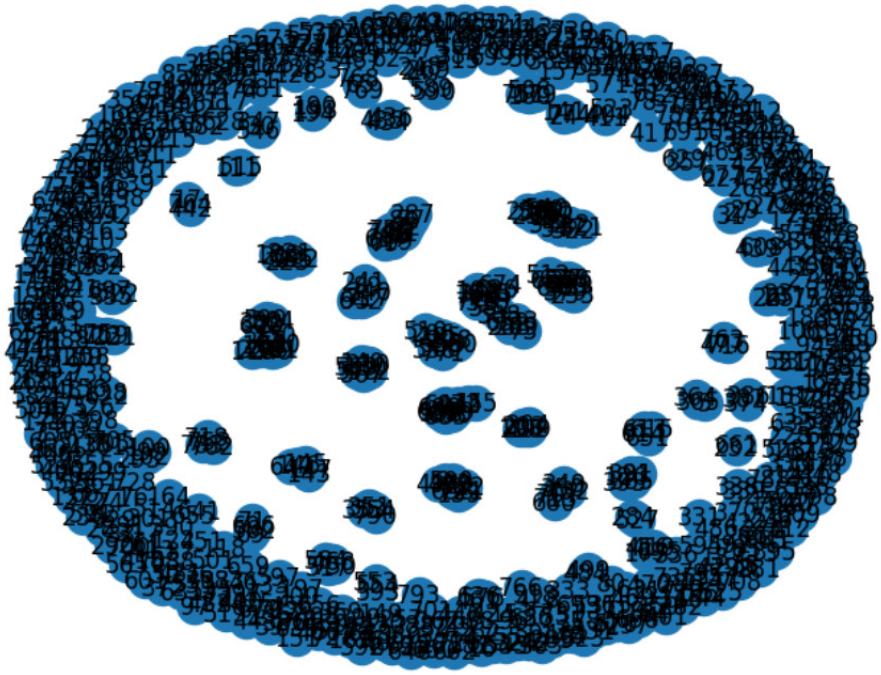
```
[24] nx.draw(g, with_labels=True)
```



```
nx.draw(G)
```



```
nx.draw(G, with_labels =True)
```



Thuật toán ComSim

```
[1] import networkx as nx
    from tqdm import tqdm
    import random
    import matplotlib.pyplot as plt
    from typing import List
```

```
[2] # BOLD_GREEN = "\x1b[38;5;2;1m"
    # RESET_CODE = "\x1b[0m"

    # class ColorFormatter(logging.Formatter):
    #     FORMAT_STR = BOLD_GREEN + "%(asctime)s - %(name)s - %(levelname)s: " + RESET_CODE + "%(message)s"
    #     def format(self, record):
    #         formatter = logging.Formatter(self.FORMAT_STR)
    #         return formatter.format(record)

    # logging.basicConfig(level=logging.INFO)
    # handlers = logging.root.handlers
    # for handler in handlers:
    #     handler.setFormatter(ColorFormatter())
    # logger = logging.getLogger()
```

```
[3] class colors:
        RED = '\033[91m'
        GREEN = '\033[92m'
        YELLOW = '\033[93m'
        # BLUE = '\033[94m'
        BLUE = '\033[36m' #'\033[34m'
        END = '\033[0m'
        VIOLET = '\033[45m'
        WHITE = '\033[47m'
        BLACK = '\033[40m'

    class styles:
      BOLD = '\033[1m' #in đậm
      ITALICS = '\033[3m' # in nghiêng
      UNDER = '\033[4m' # gạch chân
      STRIKE = '\033[9m' # gạch ngang
```

```
[4] # Định nghĩa 3.1. Vùng lân cận của đỉnh u
    def find_neighborhood(G: nx.Graph, u):
      '''Tìm các lân cận của đỉnh u trong đồ thị G

      Input:
        - G (nx.Graph): Đồ thị 2 phía G
        - u: Đỉnh u của đồ thị G

        Output:
          - N_u (set): Tập hợp các đỉnh lân cận của u trong G
      '''
      # N_u = set()
      # for v in list(G.nodes()):
      #   if G.has_edge(u,v):
      #     N_u.add(v)

      N_u = set()
      for node in G.neighbors(u):
        N_u.add(node)
      return N_u
```

```python
# Định nghĩa 3.2. (Lân cận bậc hai của đỉnh u và đỉnh v)
def find_lv2_neighborhood(G: nx.Graph, u)->set:
  '''Tìm các lân cận bậc 2 của đỉnh u trong đồ thị G

  Input:
    - G (nx.Graph): Đồ thị 2 phía G
    - u: Đỉnh u của đồ thị G

    Output:
      - N_u (set): Tập hợp các đỉnh lân cận của u trong G
  '''
  # N_u = set()
  # for v in list(G.nodes()):
  #   if G.has_edge(u,v):
  #     N_u.add(v)

  N_u = find_neighborhood(G, u)
  NN_u = set()
  for y in N_u:
    N_y = find_neighborhood(G, y)
    for x in N_y:
      NN_u.add(x)
  return NN_u - {u}
```

```python
def theta(G, x, y)->int:
  '''Hàm theta
  '''
  N_x = find_neighborhood(G, x)
  N_y = find_neighborhood(G, y)
  return len(N_x.intersection(N_y))
```

```python
def create_projection_graph(G: nx.Graph(), U: List, theta = theta)-> nx.Graph():
  """Hàm tạo đồ thị hình chiếu số đỉnh lân cận chung của các đỉnh thuộc U của G

  Input:
    - G (nx.Graph): Đồ thị 2 phía G ban đầu
    - U (List): Tập các đỉnh U của G

  Output:
    - P (nx.Graph): Đồ thị hình chiếu biểu thị số đỉnh lân cận chung của các đỉnh thuộc U
  """
  P = nx.Graph()
  P.add_nodes_from(U)
  for idx, node_1 in enumerate(U[:-1]):
    for node_2 in U[idx+1:]:
      _weight = theta(G, node_1, node_2)
      if _weight == 0:
        continue
      P.add_edge(node_1, node_2, weight=_weight)
  return P
```

```python
def find_cycle(G: nx.Graph(), H: set, y, x)->set:
  '''Trích xuất chu trình được phát hiện từ y đến x trong H
  '''
  print(f'{colors.YELLOW} Trích xuất chu trình được phát hiện từ {colors.BLUE}{y} {colors.YELLOW}đến {colors.BLUE}{x} {colors.YELLOW}trong {colors.BLUE}{H} {colors.END}')

  # Kiểm tra tính liên thông giữa 2 điểm x, y trong G
  H_graph = G.subgraph(H)
  # has_path = nx.has_path(H, y, x)
  # if has_path:
  #   shortest_path = nx.shortest_path(G, y, x)
  shortest_path = nx.shortest_path(G, y, x)
  print(f'{colors.BLUE}C{colors.YELLOW} = {colors.BLUE}{shortest_path}{colors.END}')
  return set(shortest_path)
```

```
[9]  # A Toy Sample Fig. 2 (a)
     B = nx.Graph()
     # Add nodes with the node attribute "bipartite"
     B.add_nodes_from(['u', 'u1', 'u2', 'u3'], bipartite=0)
     B.add_nodes_from(["v", "v1", "v2"], bipartite=1)
     # Add edges only between nodes of opposite node sets
     B.add_edges_from([('v1', 'u'), ('v2', 'u'), ('u', 'v'), ('v', 'u1'), ('v', 'u2'), ('v', 'u3')])
```

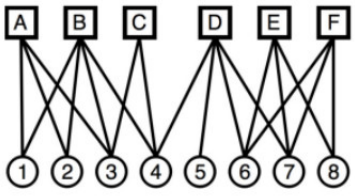
```
[10] def algorithm_3(G: nx.Graph, theta = theta):
       '''COMSIM Bước thứ nhất
       Input:
         - G (nx.Graph): Đồ thị hai phần G=(U, V, E)
         - theta (Callable): Hàm tương tự theta
       Output:
         - P (set): phân vùng P gồm các đỉnh U và một tập hợp K các nút còn lại cho bước thứ 2
       '''
       # P = set() # tập hợp phân vùng
       P = []
       U = [node for node, attr in B.nodes(data=True) if attr['bipartite'] == 0]
       # V = [node for node, attr in B.nodes(data=True) if attr['bipartite'] == 1]
       T = set(U) # tập hợp các đỉnh được xem xét

       projection_graph = create_projection_graph(G, U)
       list_nodes = list(T)
       # x = random.choice(list_nodes) # đỉnh ngẫu nhiên
       # T = T - {x}
```

```
     x = list_nodes[0]
     max_value_theta = 0
     for node in list_nodes:
       if len(find_lv2_neighborhood(G, node)) > max_value_theta:
         x = node
         max_value_theta = len(find_lv2_neighborhood(G, node))

     H = set() # tập hợp các đỉnh hiện đang được xem xét
     K = set() # tập hợp các nút còn lại

     while len(T) != 0:
       print(f'{styles.BOLD}{colors.YELLOW}---*---START WHILE LOOP---*---{colors.END}')
       print(f'Xét node: {colors.BLUE}{x}{colors.END}')
       # /* tìm một đỉnh lân cận y ∈ N(N(x)) của x mà hàm θ(x, y) đạt giá trị lớn nhất.
       NN_x = find_lv2_neighborhood(B, x)
       print(f'Danh sách lân cận bậc 2 của {colors.BLUE}{x}{colors.END}:\n {colors.BLUE}{NN_x}{colors.END}')
       if NN_x is None:
         list_nodes = list(T)
         x = random.choice(list_nodes) # đỉnh ngẫu nhiên
         T = T- {x}
         continue
       max_theta = 0
       y = x
       for _y in NN_x:
         if theta(B, x, _y) > max_theta:
           max_theta = theta(B, x, _y)
           y = _y
       print(f'Find node max theta (y): {colors.BLUE}{y}{colors.END}')
       print(f'Tập hợp các đỉnh được xem xét (H):\n \t{colors.BLUE}{H}{colors.END}')
       print(f'Danh sách các đỉnh đang được xem xét (T):\n{colors.BLUE}{T}{colors.END}')
```

```
      if y in H:
        print('---> Case: y in H <---')
        C = find_cycle(projection_graph, H, y, x)
        # for node in C:
        #   P.add(node)
        P.append(C)
        K = K | (H-C)
        H = set()
        list_nodes = list(T)
        x = random.choice(list_nodes) # đỉnh ngẫu nhiên
        T.remove(x)
      else:
        print('---> Case: y not in H <---')
        if y in T:
          print(f'\t ---> Case: y in T <---')
          H.add(y)
          x = y
          T = T - {y}
        else:
          print(f'\t ---> Case: y not in T <---')
          # /* y đã là một phần tử của P, các đỉnh đã truy cập được lưu trữ
          K = K | H
          H = set()
          list_nodes = list(T)
          x = random.choice(list_nodes) # đỉnh ngẫu nhiên
          T = T - {x}
        print(f'{colors.GREEN} Update H, T{colors.END}')
        print(f'Tập hợp các đỉnh được xem xét (H):\n \t {colors.BLUE}{H}{colors.END}')
        print(f'Danh sách các đỉnh đang được xem xét (T):\n \t{colors.BLUE}{T}{colors.END}')

    print(f'{styles.BOLD}{colors.GREEN}--End of Algorithm -- \n{colors.BLUE}P: {P},\nK: {K}{colors.END}')
    return P, K
```

```
[11] def algorithm_4(G: nx.Graph, P, K, theta):
       '''COMSIM Bước thứ hai
       Input:
         - G (nx.Graph): đồ thị hai phần G = (U, V, E)
         - P: Một phân vùng P,
         - K:  Một tập hợp K các đỉnh dư lại sau (bước đầu tiên)
         - θ: hàm tương tự theta (θ)
       Output:
         - _P: Phâm vùng P' gồm các đỉnh U và các đỉnh không thỏa mãn R
       '''
       R = set()
       _p = P
       for x in K:
         # Px = comneigh(x, P) // Tìm tất cả các cộng đồng lân cận của x
         P_x = set()
         if len(P_x) == 0:
           R = R.add(x)
         else:
           # C := argmax CxϵPx PyϵCx θ(x, y)
           C.add(x) # ???
```

A Sample

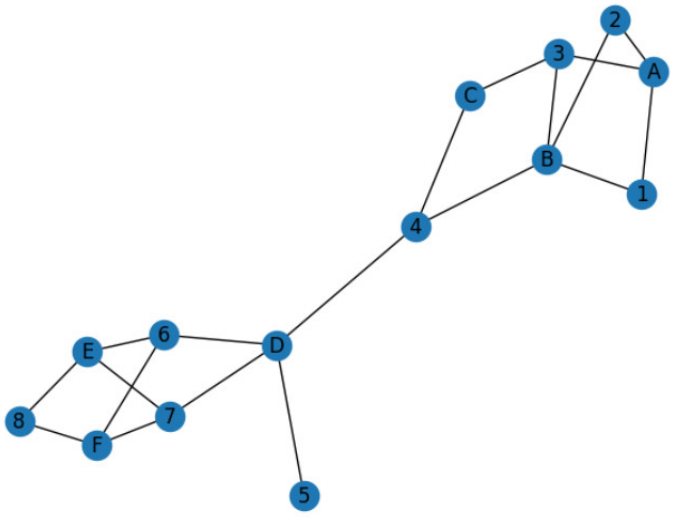


```
[12] B = nx.Graph()
     # Add nodes with the node attribute "bipartite"
     B.add_nodes_from(['A', 'B', 'C', 'D', 'E', 'F'], bipartite=0)
     B.add_nodes_from(['1', '2', '3', '4', '5', '6', '7', '8'], bipartite=1)
     # Add edges only between nodes of opposite node sets
     B.add_edges_from([('1', 'A'), ('1', 'B'),
                       ('2', 'A'), ('2', 'B'),
                       ('3', 'A'), ('3', 'B'), ('3', 'C'),
                       ('4', 'B'), ('4', 'C'), ('4', 'D'),
                       ('5', 'D'),
                       ('6', 'D'), ('6', 'E'), ('6', 'F'),
                       ('7', 'D'), ('7', 'E'), ('7', 'F'),
                       ('8', 'E'), ('8', 'F')])
```

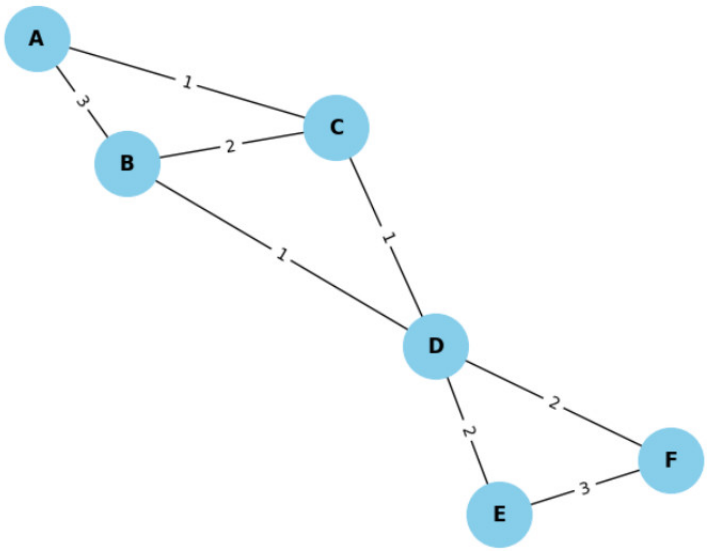
```
[13] bipartite_0_nodes = [node for node, attr in B.nodes(data=True) if attr['bipartite'] == 0]

     print(bipartite_0_nodes)
```

```
['A', 'B', 'C', 'D', 'E', 'F']
```

```
[14] nx.draw(B, with_labels=True)
```




```
[15] U = [node for node, attr in B.nodes(data=True) if attr['bipartite'] == 0]
     P = create_projection_graph(B, U)

     # nx.draw(P, with_labels=True)
     pos = nx.spring_layout(P)
     nx.draw(P, pos, with_labels=True, node_size=1500, node_color='skyblue', font_size=12, font_color='black', font_weight='bold')

     # Lấy thông tin trọng số của các cạnh
     edge_labels = nx.get_edge_attributes(P, 'weight')

     # Vẽ trọng số của các cạnh
     nx.draw_networkx_edge_labels(P, pos, edge_labels=edge_labels)

     # Hiển thị đồ thị
     plt.show()
```




```
[16] algorithm_3(B, theta = theta)
```

```
---*---START WHILE LOOP---*---
Xét node: D
Danh sách lân cận bậc 2 của D:
 {'E', 'C', 'F', 'B'}
Find node max theta (y): E
Tập hợp các đỉnh được xem xét (H):
        set()
Danh sách các đỉnh đang được xem xét (T):
{'E', 'F', 'B', 'C', 'A', 'D'}
---> Case: y not in H <---
         ---> Case: y in T <---
 Update H, T
Tập hợp các đỉnh được xem xét (H):
         {'E'}
Danh sách các đỉnh đang được xem xét (T):
        {'C', 'B', 'F', 'A', 'D'}
---*---START WHILE LOOP---*---
Xét node: E
Danh sách lân cận bậc 2 của E:
 {'F', 'D'}
Find node max theta (y): F
Tập hợp các đỉnh được xem xét (H):
        {'E'}
Danh sách các đỉnh đang được xem xét (T):
{'C', 'B', 'F', 'A', 'D'}
---> Case: y not in H <---
         ---> Case: y in T <---
 Update H, T
Tập hợp các đỉnh được xem xét (H):
         {'E', 'F'}
Danh sách các đỉnh đang được xem xét (T):
        {'C', 'A', 'B', 'D'}
---*---START WHILE LOOP---*---
```

```
Xét node: F
Danh sách lân cận bậc 2 của F:
 {'E', 'D'}
Find node max theta (y): E
Tập hợp các đỉnh được xem xét (H):
        {'E', 'F'}
Danh sách các đỉnh đang được xem xét (T):
{'C', 'A', 'B', 'D'}
---> Case: y in H <---
 Trích xuất chu trình được phát hiện từ E đến F trong {'E', 'F'}
C = ['E', 'F']
---*---START WHILE LOOP---*---
Xét node: C
Danh sách lân cận bậc 2 của C:
 {'A', 'B', 'D'}
Find node max theta (y): B
Tập hợp các đỉnh được xem xét (H):
        set()
Danh sách các đỉnh đang được xem xét (T):
{'A', 'B', 'D'}
---> Case: y not in H <---
          ---> Case: y in T <---
 Update H, T
Tập hợp các đỉnh được xem xét (H):
         {'B'}
Danh sách các đỉnh đang được xem xét (T):
        {'A', 'D'}
---*---START WHILE LOOP---*---
Xét node: B
Danh sách lân cận bậc 2 của B:
 {'C', 'A', 'D'}
Find node max theta (y): A
Tập hợp các đỉnh được xem xét (H):
        {'B'}
Danh sách các đỉnh đang được xem xét (T):
{'A', 'D'}
---> Case: y not in H <---
          ---> Case: y in T <---
 Update H, T
```

```
 Tập hợp các đỉnh được xem xét (H):
          {'A', 'B'}
 Danh sách các đỉnh đang được xem xét (T):
         {'D'}
 ---*---START WHILE LOOP---*---
 Xét node: A
 Danh sách lân cận bậc 2 của A:
  {'C', 'B'}
 Find node max theta (y): B
 Tập hợp các đỉnh được xem xét (H):
         {'A', 'B'}
 Danh sách các đỉnh đang được xem xét (T):
 {'D'}
 ---> Case: y in H <---
  Trích xuất chu trình được phát hiện từ B đến A trong {'A', 'B'}
 C = ['B', 'A']
 --End of Algorithm --
 P: [{'E', 'F'}, {'A', 'B'}],
 K: set()
 ([{'E', 'F'}, {'A', 'B'}], set())
```