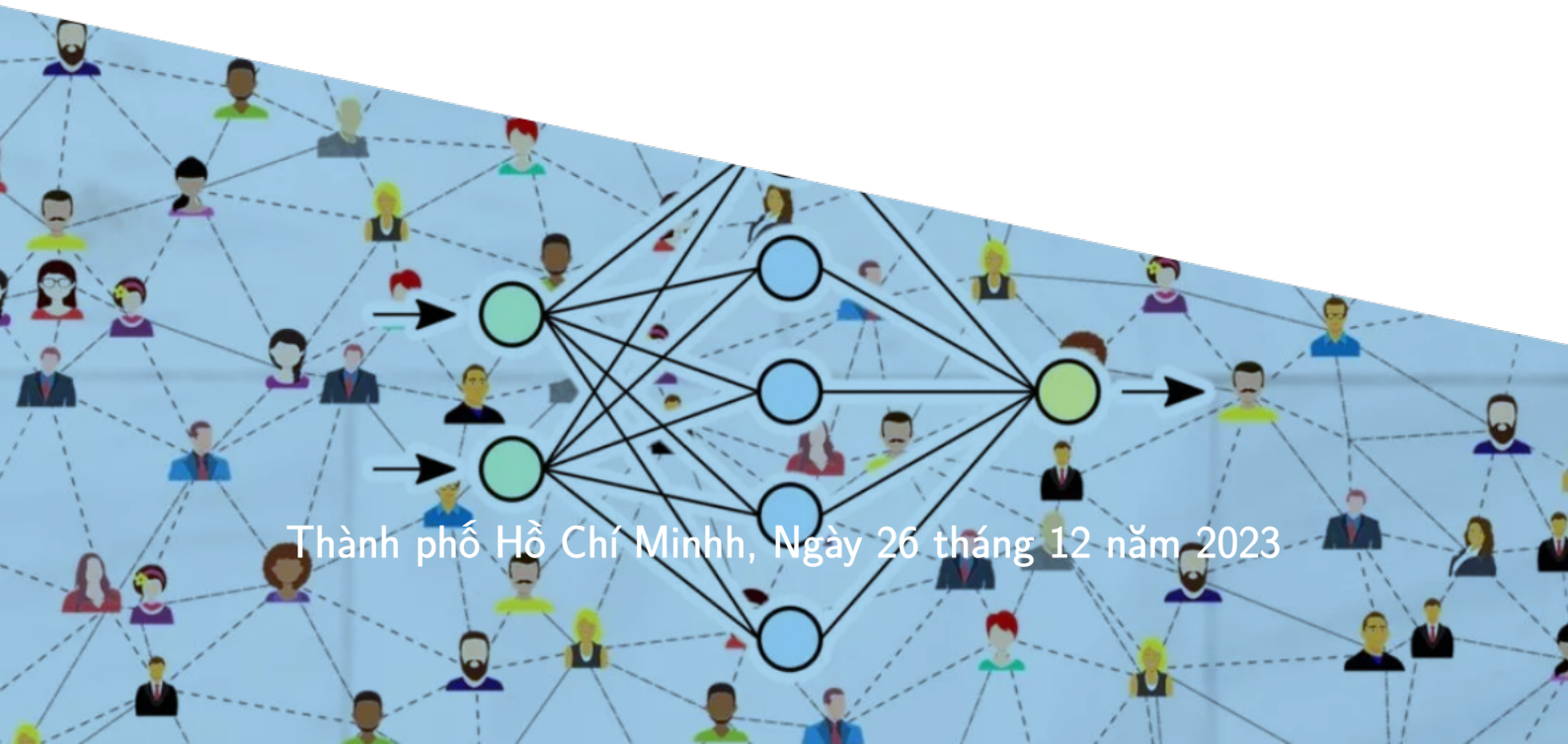
# Vấn đề đồ thị con đẳng cấu trong phân tích mạng xã hội

CO1007 - Cấu trúc rời rạc - HK231


**Lớp L01 - Nhóm Hứa sẽ qua môn**

| Full Name | Student ID |
|---|---|
| Trần Như Mai Anh | 2210140 |
| Ngô Đức Luân | 2211949 |
| Tạ Quang Thái | 2213117 |
| Huỳnh Quang Thịnh | 2213273 |
| Đậu Ngọc Quân | 2212777 |

Giảng viên hướng dẫn: Nguyễn An Khương

Thành phố Hồ Chí Minhh, Ngày 26 tháng 12 năm 2023

# Mục lục


**1 Mở đầu** **1**

**2 Cơ sở lý thuyết** **2**
2.1 Đồ thị . . . . . 2
2.2 Biểu diễn đồ thị . . . . . 3
2.3 Tính thẩm mỹ trong việc trực quan hóa đồ thị . . . . . 10
2.4 Đồ thị đẳng cấu và đồ thị con đẳng cấu . . . . . 13
2.5 Tam giác, k-truss, và ý nghĩa của chúng đối với đồ thị bạn bè . . . . . 14
2.6 Ý nghĩa bài toán đếm tam giác và phân rã truss đối với bài toán đồ thị đẳng cấu . . . . . 16

**3 Hướng tiếp cận** **17**
3.1 Đếm tam giác . . . . . 17
3.2 Phân rã truss . . . . . 18
3.3 Xử lý dữ liệu . . . . . 20
3.4 Trực quan hóa đồ thị . . . . . 21

**4 Thực nghiệm** **22**
4.1 Thực nghiệm kiểm thử tính đúng đắn của thuật toán trên dataset có số lượng nhỏ . . . . . 22
4.2 Áp dụng thuật toán trên dataset thực tế từ mạng xã hội Facebook . . . . . 24

**5 Kết luận** **28**

**6 Tài liệu** **29**

# 1 | Mở đầu

Đồ thị là một dạng cấu trúc dữ liệu gồm tập hợp các cạnh và các đỉnh và được dùng để biểu diễn mối quan hệ giữa các cá thể. Đồ thị được ứng dụng rộng rãi nhằm biểu diễn,lưu trữ, tìm kiếm, và tối ưu hóa trong các nhiều lĩnh vực khác nhau. Lý thuyết đồ thị là một ngành rộng lớn với nhiều bài toán khác nhau được đặt ra như bài toán tô màu, bài toán đường đi, bài toán phủ. Trong phạm vi bài báo cáo này, chúng ta sẽ đi tìm hiểu về bài toán đồ thị con đẳng cấu, một trong những bài toán thường gặp và có ứng dụng rộng rãi.

Bài toán phát hiện đồ thị con đẳng cấu có đầu vào đồ thị G và H được cho trước và yêu cầu xác định đồ thị G có chứa đồ thị con nào đẳng cấu với đồ thị H hay không. Bài toán là trường hợp tổng quát hóa của bài toán xác định chu trình Hamilton và Clique tối đa. Bài toán được ứng dụng rộng rãi trong các lĩnh vực mạch điện tử, sinh hóa học, trí tuệ nhân tạo và trong việc phân tích mạng xã hội.

Mạng xã hội, tiếng Anh là social network hoặc social media network, được định nghĩa là "hệ thống thông tin cung cấp cho cộng đồng người dùng mạng các dịch vụ lưu trữ, cung cấp, sử dụng, tìm kiếm, chia sẻ và trao đổi thông tin với nhau"[1]. Việc phân tích mạng xã hội được thực hiện chủ yếu dựa trên lý thuyết đồ thị. Thông thường, mạng xã hội sẽ được mô hình hóa thành đồ thị vô hướng (bao gồm đồ thị tình bạn, đồ thị tương tác) và đồ thị có hướng (bao gồm đồ thị tương tác ngầm, đồ thị theo dõi). Trong đó đồ thị tình bạn bao gồm các đỉnh biểu diễn các cá thể (individual actor) và các cạnh biểu diễn mối quan hệ bạn bè (relationship).

Trong thời đại công nghệ thông tin phát triển, khi mà mạng xã hội ngày càng lớn mạnh, việc phân tích mạng xã hội ngày càng được chú trọng và có nhiều ảnh hưởng đến các lĩnh vực khác nhau như thị trường kinh tế, chính trị, an ninh xã hội và hơn thế nữa. Một trong những vấn đề được quan tâm nhất có thể kể đến việc phát hiện các cộng đồng và phân tích độ kết nối trong các cộng đồng thông qua bài toán đếm tam giác (triangle counting) và phân rã đồ thị truss (truss decomposition) dựa trên mô hình đồ thị bạn bè của mạng xã hội. Hai bài toán trên đều là hai bài toán nhỏ thuộc bài toán đồ thị con đẳng cấu được nêu trên.

Bài nghiên cứu này sẽ trình bày các cơ sở lý thuyết của bài toán và đưa ra hướng giải quyết vấn đề bằng thuật toán cụ thể. Sau đó thuật toán sẽ được áp dụng cho dataset thực tế từ mạng xã hội và được phân tích nhằm giúp ta hiểu hơn về cách giải quyết bài toán đếm tam giác và phân rã truss và ứng dụng của chúng nói riêng và lý thuyết đồ thị nói chung trong việc phân tích mạng xã hội trong thực tế.

[1] Văn bản hợp nhất 02/VBHN-BTTTT 2018

# 2 | Cơ sở lý thuyết

## 2.1 | Đồ thị

- **Đồ thị**

  Một đồ thị là một cấu trúc rời rạc gồm tập hợp các đỉnh và các cạnh nối giữa các đỉnh đó. Có thể mô tả đồ thị theo cách hình thức như sau: G=(V,E), trong đó V là tập đỉnh khác rỗng và E là tập các cạnh.

  Có thể phân loại đồ thị G theo tính chất của tập cạnh như sau:

  - Đồ thị vô hướng
  - Đồ thị có hướng
  - Đồ thị lai
  - Đa đồ thị
  - Đơn đồ thị

- **Đồ thị vô hướng**

  Là một cặp G=(V, E), trong đó:

  - V: tập các đỉnh hoặc nút.
  - E: tập các cặp không thứ tự chứa các đỉnh phân biệt, được gọi là cạnh. Hai đỉnh thuộc một cạnh được gọi là các điểm đầu mút của cạnh đó.

  **Các định nghĩa trên đồ thị vô hướng**

  Cho đồ thị vô hướng G = (V,E), trong đó V là tập đỉnh, E là tập cạnh.

  - Định nghĩa 1. Hai đỉnh u và v của đồ thị vô hướng G được gọi là kề nhau nếu {u,v} là cạnh thuộc đồ thị G. Nếu e ={u, v} là cạnh của đồ thị G thì ta nói cạnh này liên thuộc với hai đỉnh u và v, hoặc ta nói cạnh e nối đỉnh u với đỉnh v, đồng thời các đỉnh u và v sẽ được gọi là đỉnh đầu của cạnh u,v.
  - Định nghĩa 2. Ta gọi bậc của đỉnh v trong đồ thị vô hướng là số cạnh liên thuộc với nó và ký hiệu là deg(v). Một đỉnh có bậc 0 là đỉnh cô lập. Đỉnh có bậc 1 là một đỉnh treo hay lá.

- **Đồ thị có hướng**

  Là một cặp G:=(V, A), trong đó:

  - V: tập các đỉnh hoặc nút.
  - A: tập các cặp có thứ tự chứa các đỉnh, được gọi là các cạnh có hướng hoặc cung. Một cạnh e = (x, y) được coi là có hướng từ x tới y; x được gọi là đỉnh đầu và y được gọi là đỉnh cuối của cạnh.

  **Các định nghĩa trên đồ thị có hướng**

  Cho đồ thị có hướng G = (V,E), trong đó V là tập đỉnh, E là tập cạnh.

  - Định nghĩa 1. Nếu e=(u,v) là cung của đồ thị có hướng G thì ta nói hai đỉnh u nối tới v, v được nối từ u, và nói cung (u, v) nối đỉnh u với đỉnh v, hoặc nói cung này đi ra khỏi đỉnh u và đi vào đỉnh v. Đỉnh u được gọi là đỉnh đầu, đỉnh v được gọi là đỉnh cuối của cung (u,v).
  - Định nghĩa 2. Ta gọi bán bậc ra của đỉnh v trên đồ thị có hướng là số cung của đồ thị đi ra khỏi v và ký hiệu là deg+ (v). Ta gọi bán bậc vào của đỉnh v trên đồ thị có hướng là số cung của đồ thị đi vào v và ký hiệu là deg- (v).

- **Đường đi và chu trình**
  - Đường đi của độ dài n từ u tới v: chuỗi liên tiếp của n cạnh $\{x_0, x_1\}, \{x_1, x_2\}, \{x_2, x_3\}, \ldots, \{x_{n-1}, x_n\}$, với $x_0 = u$ và $x_n = v$.
  - Chu trình trong đồ thị là một đường đi đóng trong đồ thị, nghĩa là đường đi đó bắt đầu và kết thúc tại 1 đỉnh, u = v.

## 2.2 | Biểu diễn đồ thị

### 2.2.1 | Biểu diễn đồ thị bằng ma trận kề

Cấu trúc dữ liệu phổ dụng nhất để biểu diễn đồ thị là biểu diễn đồ thị bằng ma trận. Về lý thuyết, người ta đã chứng minh được mỗi ma trận vuông (0,1) cấp n đều đẳng cấu với một đơn đồ thị vô hướng hoặc có hướng. Mục này, chúng ta sẽ xem xét phương pháp biểu diễn các loại đồ thị khác nhau bằng ma trận kề.

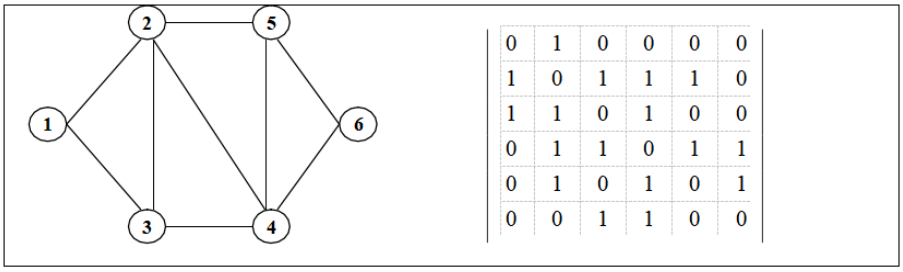
**Ma trận kề của đồ thị vô hướng**

Xét đồ thị đơn vô hướng G =(V, E), với tập đỉnh$V = \{1, 2, ..., n\}$, tập cạnh $E = \{e1, e2, .., em\}$. Ta gọi ma trận kề của đồ thị G là ma trận có các phần tử hoặc bằng 0 hoặc bằng 1 theo qui định như sau:

$$A = \{a_{ij} : a_{ij} = 1 \ nếu \ (i, j) \in E, a_{ij} = 0 \ nếu \ (i, j) \notin E; i, j = 1, 2, \ldots, n\}.$$

Ví dụ 1. Biểu diễn đồ thị trong Hình 2.1 dưới đây bằng ma trận kề.

Tính chất ma trận kề đối với đồ thị vô hướng:




$$\begin{vmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{vmatrix}$$

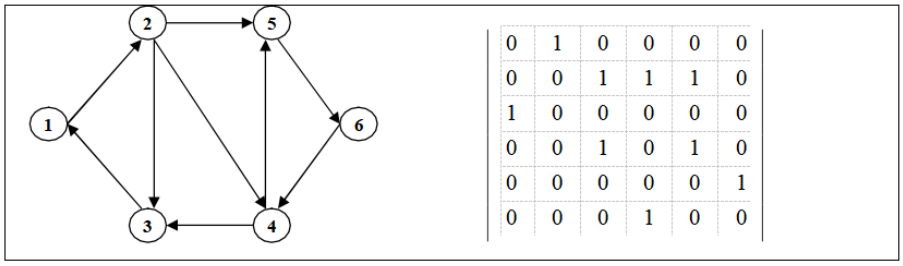
**Hình 2.1:** Ma trận kề biểu diễn đồ thị vô hướng

1. Tổng các phần tử của ma trận bằng hai lần số cạnh : $\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} = 2m$(m là số cạnh của đồ thị).
2. Tổng các phần tử của hàng u là bậc của đỉnh u: $deg(u) = \sum_{j=1}^{n} a_{uj}$ .Ví dụ với ma trận kề biểu diễn đồ thị Hình 2.1, tổng các phần tử của hàng 1 là bậc của đỉnh 1, vì vậy deg(1)=2; tổng các phần tử của hàng 2 là bậc của đỉnh 2, vì vậy deg(2)=3.
3. Tổng các phần tử của cột u là bậc của đỉnh u:$deg(u) = \sum_{j=1}^{n} a_{ju}$ .Ví dụ với ma trận kề biểu diễn đồ thị Hình 2.1, tổng các phần tử của cột 1 là bậc của đỉnh 1, vì vậy deg(1)=2; tổng các phần tử của cột 2 là bậc của đỉnh 2, vì vậy deg(2)=3.
4. Nếu ký hiệu $a_{ij}^{p}$ với $i, j = 1, 2, \ldots, n$ là các phần tử của ma trận $A^p = A \times A \times A \times \ldots \times A$ (p lần). Khi đó,$a_{ij}^{p}$, $i, j = 1, 2, \ldots, n$ cho ta số đường đi khác nhau từ đỉnh i đến đỉnh j qua p-1 đỉnh trung gian.

**Ma trận kề của đồ thị có hướng**

Ma trận kề của đồ thị có hướng cũng được định nghĩa hoàn toàn tương tự, chúng ta chỉ cần lưu ý tới hướng của cạnh. Ma trận kề của đồ thị có hướng là không đối xứng.

Ví dụ 2. Ma trận kề của đồ thị có hướng trong Hình 2.2.




**Hình 2.2:** Ma trận kề của đồ thị có hướng.

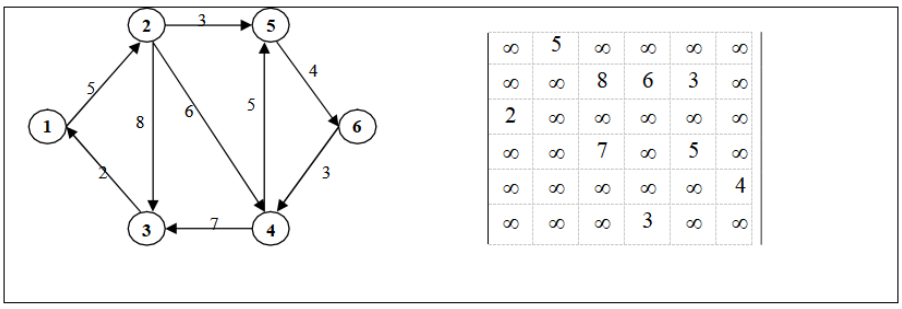
Tính chất của ma trận kề của đồ thị có hướng:

1. Tổng các phần tử của ma trận bằng số cạnh $\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} = m$(m là số cạnh của đồ thị).
2. Tổng các phần tử của hàng u là bán đỉnh bậc ra của đỉnh u:$deg^{+}(u) = \sum_{j=1}^{n} a_{uj}$ .Ví dụ với ma trận kề biểu diễn đồ thị Hình 2.2, tổng các phần tử của hàng 1 là bán đỉnh bậc a của đỉnh 1, vì vậy $\deg^{+}(1)$=1; tổng các phần tử của hàng 2 là bán đỉnh bậc ra của đỉnh 3, vì vậy $\deg^{+}(2)$=3.
3. Tổng các phần tử của cột u là bán đỉnh bậc vào của đỉnh u:$deg^{-}(u) = \sum_{j=1}^{n} a_{ju}$ .Ví dụ với ma trận kề biểu diễn đồ thị Hình 2.2, tổng các phần tử cột 1 là bán đỉnh bậc vào của đỉnh 1, vì vậy $\deg^{-}(1)$=1; tổng các phần tử của cột 2 là bán đỉnh bậc vào của đỉnh 2, vì vậy $\deg^{-}(2)$=1.
4. Nếu ký hiệu $a_{ij}^{p}$, $i, j = 1, 2, \ldots, n$ là các phần tử của ma trận $A^p = A \times A \times A \times \ldots \times A$ (p lần). Khi đó,$a_{ij}^{p}$, $i, j = 1, 2, \ldots, n$ cho ta số đường đi khác nhau từ đỉnh i đến đỉnh j qua p-1 đỉnh trung gian.

**Ma trận trọng số**

Trong rất nhiều ứng dụng khác nhau của lý thuyết đồ thị, mỗi cạnh e =(u,v) của nó được gán bởi một số c(e) = c(u,v) gọi là trọng số của cạnh e. Đồ thị trong trường hợp như vậy gọi là đồ thị trọng số. Trong trường hợp đó, ma trận kề của đồ thị được thay bởi ma trận trọng số c= $c_{ij}$, $i, j = 1, 2, \ldots, n$ . $c_{ij} = c(i, j)$ nếu (i, j) ∈E, $c_{ij} = \theta$ nếu (i, j) ∉E. Trong đó, $\theta$ nhận các giá trị: 0,$\infty$, $-\infty$ tuỳ theo từng tình huống cụ thể của thuật toán.

Ví dụ 3. Ma trận kề của đồ thị có trọng số trong Hình 2.3.




**Hình 2.3:** Ma trận kề của đồ thị có trọng số

**Ưu điểm của ma trận kề**

- Đơn giản dễ cài đặt trên máy tính bằng cách sử dụng một mảng hai chiều để biểu diễn ma trận kề.

- Dễ dàng kiểm tra được hai đỉnh u, v có kề với nhau hay không bằng đúng một phép so sánh ($a_{uv} \neq 0$?);và chúng ta chỉ mất đúng một phép so sánh.

**Nhược điểm ma trận kề**

- Lãng phí bộ nhớ: bất kể số cạnh nhiều hay ít ta cần $n^2$ đơn vị bộ nhớ để biểu diễn.
- Không thể biểu diễn được với các đồ thị có số đỉnh lớn (ví dụ triệu đỉnh).
- Để xem xét đỉnh đỉnh u có những đỉnh kề nào cần mất n phép so sánh kể cả đỉnh u là đỉnh cô lập hoặc đỉnh treo.

### 2.2.2 | Biểu diễn đồ thị bằng ma trận liên thuộc

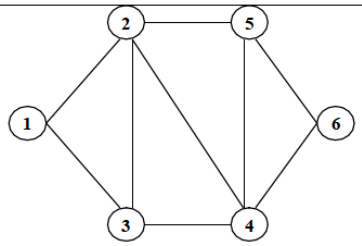
Ma trận liên thuộc là một cách biểu diễn đồ thị bằng ma trận quen thuộc và nó mã hóa mối quan hệ giữa cạnh và đỉnh.

**Ma trận liên thuộc của đồ thị vô hướng**

Xét đồ thị đơn vô hướng G =(V, E), với tập đỉnh$V = \{1, 2, ..., n\}$, tập cạnh $E = \{e1, e2, .., em\}$. Ta gọi ma trận liên thuộc của đồ thị G là ma trận có kích thước bằng $n \times m$, với n,m lần lượt là số đỉnh và số cạnh của G, và có giá trị các phần tử hoặc theo qui định như sau:

$$B_{ij} = \begin{cases} 1 \text{ nếu đỉnh } v_i \text{ liên thuộc với cạnh } e_j \\ 0 \text{ nếu ngược lại} \end{cases}$$

Ví dụ 4. Ma trận liên thuộc của đồ thị vô hướng trong Hình 2.4.




$$\begin{vmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

**Hình 2.4:** Ma trận liên thuộc của đồ thị vô hướng

Tính chất ma trận liên thuộc đối với đồ thị vô hướng:

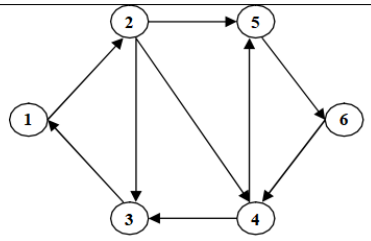
1. Tổng các phần tử mỗi cột của ma trận bằng 2, bởi mỗi cạnh được nối với 2 đỉnh.
2. Mối liên hệ giữa ma trận liên thuộc và ma trận kề: $BB^T = A + D$. Với B là ma trận liên thuộc, A là ma trận kề, D là ma trận chéo và mỗi phần tử trên đường chéo bằng với bậc của đỉnh tương ứng.

**Ma trận liên thuộc của đồ thị có hướng**

Ma trận có kích thước bằng $n \times m$, với n,m lần lượt là số đỉnh và số cạnh, và có giá trị các phần tử hoặc theo qui định như sau:

$$B_{ij} = \begin{cases} -1 \text{ nếu cạnh } e_j \text{ ra đỉnh } v_i \\ 1 \text{ nếu cạnh } e_j \text{ vào đỉnh } v_i \\ 0 \text{ nếu ngược lại} \end{cases}$$

Ví dụ 5. Ma trận liên thuộc của đồ thị có hướng trong Hình 2.5.

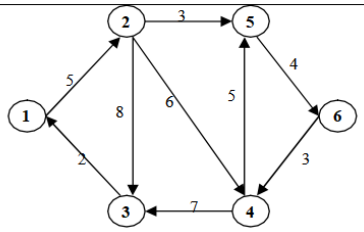



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| −1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | −1 | 0 | 0 | 0 | 0 | -1 | -1 | 0 |
| 0 | 0 | 0 | 0 | 1 | −1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | −1 | 0 | 0 | 1 | −1 |
| 0 | 1 | −1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | −1 | 0 | 0 | 0 | 0 | 0 |

**Hình 2.5:** Ma trận liên thuộc của đồ thị có hướng

**Ma trận trọng số**

Trong trường hợp đồ thị có trọng số, ta thay số 1 bằng trọng số của cạnh đó.
Ví dụ 6. Ma trận liên thuộc của đồ thị có hướng trong Hình 2.6.




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| −5 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| 5 | −3 | 0 | 0 | 0 | 0 | -8 | -6 | 0 |
| 0 | 0 | 0 | 0 | 7 | −2 | 8 | 0 | 0 |
| 0 | 0 | 0 | 3 | −7 | 0 | 0 | 6 | −5 |
| 0 | 3 | −4 | 0 | 0 | 0 | 0 | 0 | 5 |
| 0 | 0 | 4 | −3 | 0 | 0 | 0 | 0 | 0 |

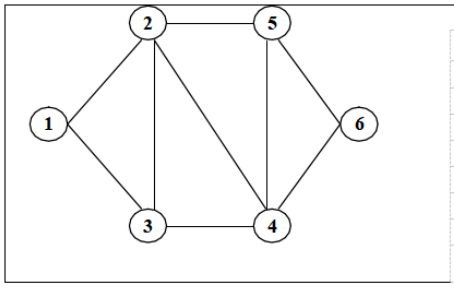
**Hình 2.6:** Ma trận liên thuộc của đồ thị có trọng số

### 2.2.3 | Biểu diễn đồ thị bằng danh sách cạnh (cung)

Trong trường hợp đồ thị thưa (đồ thị có số cạnh m $\leq$ 6n), người ta thường biểu diễn đồ thị dưới dạng danh sách cạnh. Trong phép biểu diễn này, chúng ta sẽ lưu trữ danh sách tất cả các cạnh (cung) của đồ thị vô hướng (có hướng). Mỗi cạnh (cung) e=(x, y) được tương ứng với hai biến dau[e], cuoi[e]. Như vậy, để lưu trữ đồ thị, ta cần 2m đơn vị bộ nhớ. Nhược điểm lớn nhất của phương pháp này là để nhận biết những cạnh nào kề với cạnh nào chúng ta cần m phép so sánh trong khi duyệt qua tất cả m cạnh (cung) của đồ thị. Nếu là đồ thị có trọng số, ta cần thêm m đơn vị bộ nhớ để lưu trữ trọng số của các cạnh.

**Biểu diễn đồ thị vô hướng bằng danh sách cạnh**

Đối với đồ thị vô hướng, mỗi cạnh là bộ không tính đến thứ tự các đỉnh. Do vậy, trong khi biểu diễn đồ thị vô hướng bằng danh sách cạnh ta chỉ cần liệt kê các cạnh {u,v} một lần. Để tránh nhầm lẫn, ta nên liệt kê các cạnh theo thứ tự tăng dần của đỉnh đầu mỗi cạnh. Trong trường hợp biểu diễn đa đồ thị vô hướng, ta bổ sung thêm một cột là số cạnh (socanh) nối giữa hai đỉnh của đồ thị. Hình 2.4 dưới đây mô tả chi tiết phương pháp biểu diễn đồ thị vô hướng bằng danh sách cạnh.




| Đỉnh đầu | Đỉnh cuối |
|---|---|
| 1 | 2 |
| 1 | 3 |
| 2 | 3 |
| 2 | 4 |
| 2 | 5 |
| 3 | 4 |
| 4 | 5 |
| 4 | 6 |
| 5 | 6 |

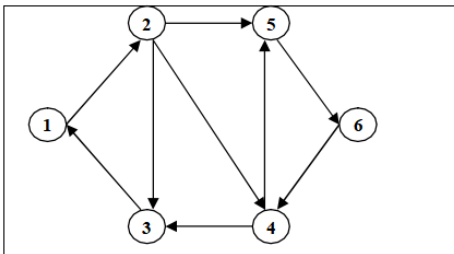
**Hình 2.7:** Danh sách cạnh của đồ thị vô hướng

Tính chất danh sách cạnh của đồ thị vô hướng:

- Đỉnh đầu nhỏ hơn đỉnh cuối mỗi cạnh.
- Số đỉnh có giá trị u thuộc cả vế phải và vế trái của danh sách cạnh là bậc của đỉnh u. Ví dụ giá trị u=1 xuất hiện 2 lần từ đó ta suy ra deg(1)=2, số 2 xuất hiện 4 lần vì vậy deg(2) = 4.

**Biểu diễn đồ thị có hướng bằng danh sách cạnh**

Trong trường hợp đồ thị có hướng, mỗi cạnh là bộ có tính đến thứ tự các đỉnh. Ví dụ cạnh (u,v) khác với cạnh (v, u). Do vậy, trong khi biểu diễn đồ thị vô hướng bằng danh sách cạnh ta đặc biệt chú ý đến hướng của các cạnh. Hình 2.5 dưới đây mô tả chi tiết phương pháp biểu diễn đồ thị có hướng bằng danh sách cạnh.




| Đỉnh đầu | Đỉnh Cuối |
|---|---|
| 1 | 2 |
| 2 | 3 |
| 2 | 4 |
| 2 | 5 |
| 3 | 1 |
| 4 | 3 |
| 4 | 5 |
| 5 | 6 |
| 6 | 4 |

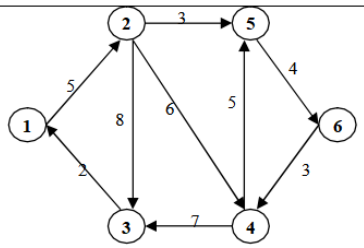
**Hình 2.8:** danh sách cạnh của đồ thị vô hướng

Tính chất danh sách cạnh của đồ thị có hướng:

- Đỉnh đầu không nhất thiết phải nhỏ hơn đỉnh cuối mỗi cạnh.
- Số đỉnh có giá trị u thuộc cả vế phải các cạnh là $deg^+(u)$. Ví dụ giá trị u=1 xuất hiện 1 lần ở vế phải của tất cả các cạnh nên $deg^+(1)$ =1, giá trị u=2 xuất hiện 3 lần ở vế phải của tất cả các cạnh nên $deg^+(2)$ =3.
- Số đỉnh có giá trị u thuộc cả vế trái các cạnh là $deg^-(u)$. Ví dụ giá trị u=1 xuất hiện 1 lần ở vế trái của tất cả các cạnh nên $deg^-(1)$ =1, giá trị u=2 xuất hiện 1 lần ở vế trái của tất cả các cạnh nên $deg^-(2)$ =1

**Biểu diễn đồ thị trọng số bằng danh sách cạnh**

Trong trường hợp đồ thị có hướng (hoặc vô hướng) có trọng số, ta bổ sung thêm một cột là trọng số của mỗi cạnh. Hình 2.6 dưới đây mô tả chi tiết phương pháp biểu diễn đồ thị trọng số bằng danh sách cạnh.




| Đỉnh đầu | Đỉnh Cuối | Trọng Số |
|---|---|---|
| 1 | 2 | 5 |
| 2 | 3 | 8 |
| 2 | 4 | 6 |
| 2 | 5 | 3 |
| 3 | 1 | 2 |
| 4 | 3 | 7 |
| 4 | 5 | 5 |
| 5 | 6 | 4 |
| 6 | 4 | 3 |

**Hình 2.9:** Biểu diễn đồ thị trọng số bằng danh sách cạnh

**Ưu điểm của danh sách cạnh:**

- Trong trường hợp đồ thị thưa (m<6n), biểu diễn bằng danh sách cạnh tiết kiệm được không gian nhớ.
- Thuận lợi cho một số thuật toán chỉ quan tâm đến các cạnh của đồ thị.

**Nhược điểm của danh sách cạnh:**

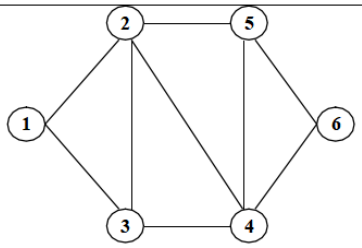
- Khi cần duyệt các đỉnh kề với đỉnh u bắt buộc phải duyệt tất cả các cạnh của đồ thị. Điều này làm cho thuật toán có chi phí tính toán cao.

### 2.2.4 | Biểu diễn đồ thị bằng danh sách kề

Trong rất nhiều ứng dụng, cách biểu diễn đồ thị dưới dạng danh sách kề thường được sử dụng. Trong biểu diễn này, với mỗi đỉnh u của đồ thị chúng ta lưu trữ danh sách các đỉnh kề với nó mà ta ký hiệu là Ke(u), nghĩa là

$$Ke(u) = \{v \in V : (u, v) \in E\}$$

Với cách biểu diễn này, mỗi đỉnh u của đồ thị, ta làm tương ứng với một danh sách tất cả các đỉnh kề với nó và được ký hiệu là List(u). Để biểu diễn List(u), ta có thể dùng các kiểu dữ liệu kiểu tập hợp, mảng hoặc danh sách liên kết. Hình 2.8 dưới đây đưa ra ví dụ chi tiết về biểu diễn đồ thị bằng danh sách kề.




Ke(1) = { 2, 3).
Ke(2) = {1, 3, 4, 5}.
Ke(3) = {1, 2, 4}.
Ke(4) = {2, 3, 5, 6}.
Ke(5) = {2, 4, 6}.
Ke(6) = { 4, 5}.

**Hình 2.10:** Biểu diễn đồ thị bằng danh sách kề

**Ưu điểm của danh sách kề:**

- Dễ dàng duyệt tất cả các đỉnh của một danh sách kề.
- Dễ dàng duyệt các cạnh của đồ thị trong mỗi danh sách kề.

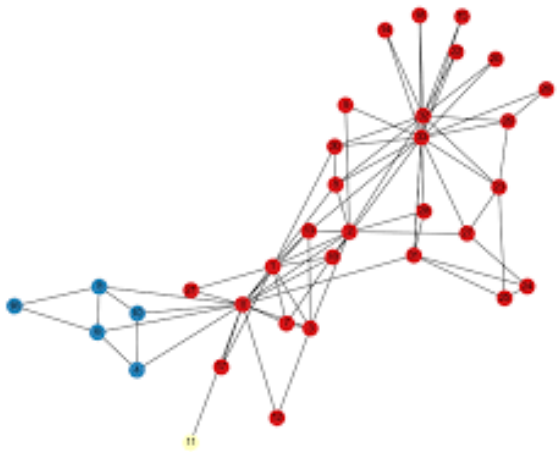
**Nhược điểm của danh sách kề:**

- Khó khăn cho người đọc có kỹ năng lập trình yếu.

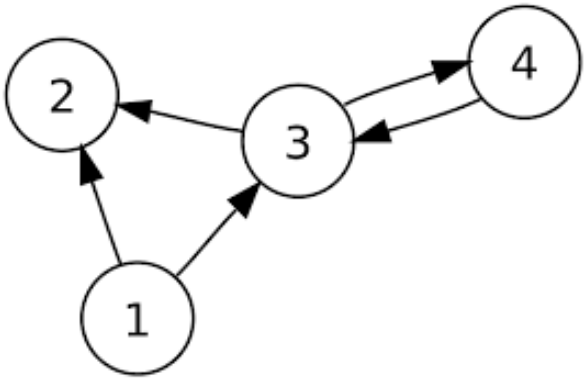
### 2.2.5 | Biểu diễn đồ thị bằng hình học

Biểu diễn đồ thị bằng hình học, hay là trực quan hóa đồ thị, là việc sử dụng các yếu tố trực quan như đường, hình dạng, màu sắc,kích thước,... để biểu diễn đồ thị, trong đó mô hình phổ biến nhất là node-link diagram, được vẽ như sau:

- Biểu diễn mỗi đỉnh của đồ thị bằng một điểm.
- Đối với đồ thị vô hướng, mỗi cạnh được biểu diễn bằng một đường cong hoặc thẳng nối giữa hai đỉnh liên thuộc với nó trong mặt phẳng.
- Đối với đồ thị có hướng, đầu mũi tên thường được sử dụng để đánh dấu hướng của cạnh.

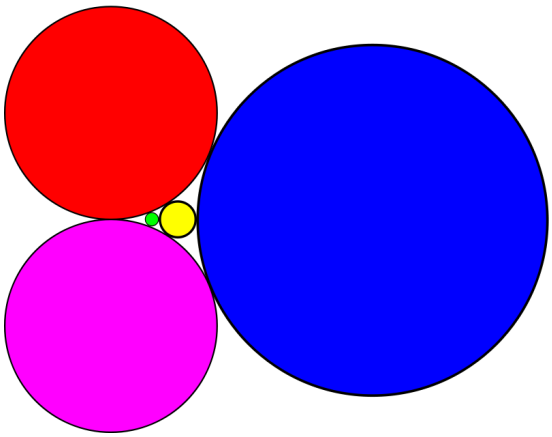


**Hình 2.11:** Biểu diễn đồ thị vô hướng bằng hình học

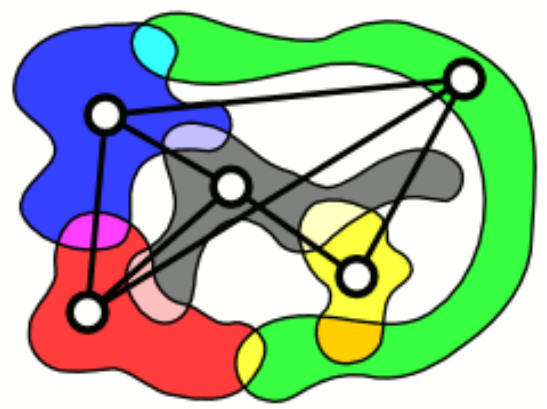



**Hình 2.12:** Biểu diễn đồ thị có hướng bằng hình học

Ngoài mô hình node-link, ta có thể bắt gặp các mô hình biểu diễn khác như mô hình circle packing, intersection representations,... và các mô hình 3D.



(a) Mô hình circle packing



(b) Mô hình intersection representations

**Ưu điểm của việc biểu diễn bằng hình học**

- Cung cấp thông tin về các mối quan hệ, mẫu (pattern), và dữ liệu ngoại lai trong đồ thị.
- Công cụ hiệu quả cho việc phân tích đồ thị.

**Nhược điểm của việc biểu diễn bằng hình học**

- Khó khăn trong việc tính toán và xử lý.
- Độ hiệu quả của phương pháp này phụ thuộc lớn vào cách đồ thị được vẽ.

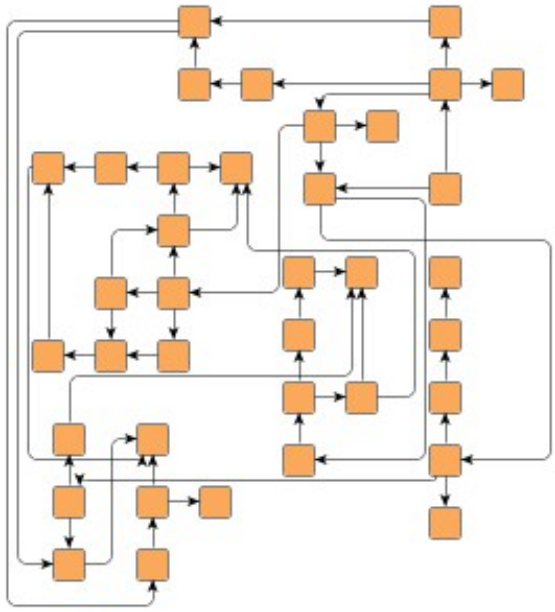
## 2.3 | Tính thẩm mỹ trong việc trực quan hóa đồ thị

Tính hiệu quả của việc trực quan hóa đồ thị phụ thuộc lớn về tính thẩm mỹ của đồ thị. Tuy không có tiêu chí nghiêm ngặt cho mọi trường hợp, những tiêu chí sau thường được xem xét khi đánh giá độ thẩm mỹ và tính hiệu quả:

- Số cạnh giao nhau là nhỏ nhất để tăng khả năng truyền tải thông tin (readability) của đồ thị và giúp người đọc dễ dàng theo dõi đường đi.
- Các đỉnh được phân bố đều trong không gian nhằm tăng thẩm mỹ và tránh việc các nút co cụm với nhau.
- Các cạnh có ít chỗ uốn cong nhất để giúp người đọc dễ dàng theo dõi đường đi của cạnh.

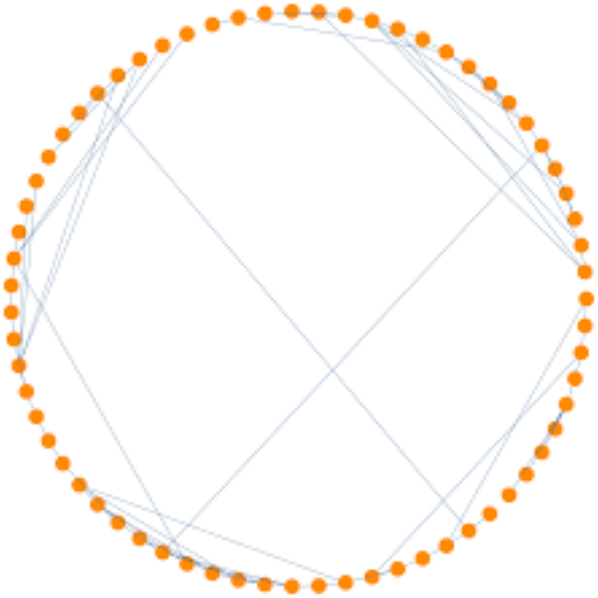
Và các tiêu chí khác tùy thuộc vào từng loại đồ thị và mục đích khác nhau. Tính thẩm mỹ của đồ thị chịu sự ảnh hưởng của nhiều yếu tố như độ lớn của đỉnh, độ dày của cạnh, màu sắc, việc đánh dấu nhãn và layout-cách các đỉnh được bố trí tương quan với nhau. Một số layout thông dụng trong việc trực quan hóa đồ thị có thể kể đến:

- **Orthogonal layout**
  Các cạnh của đồ thị chạy theo chiều ngang hoặc chiều dọc, song song với các trục tọa độ của bố cục. Orthogonal layout phù hợp với các biểu đồ thưa thớt cỡ trung bình với đặc tính nhỏ gọn, không có sự chồng chéo, ít đường giao nhau và ít chỗ uốn cong.



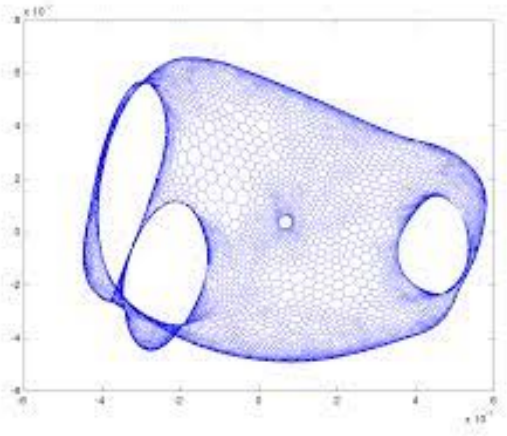
**Hình 2.14:** Orthogonal layout

- **Circular layout**
  Các đỉnh được bố trí trên một hoặc nhiều vòng tròn và được sắp xếp sao cho số cạnh giao nhau là ít nhất và các đỉnh kề nhau sẽ nằm gần nhau. Các cạnh có thể được biểu diễn thành các dây cung hoặc vòng cung trong hoặc ngoài vòng tròn. Với việc bố trí các đỉnh ở các khoảng cách đều nhau, circular layout giúp việc phân tích đồ thị trung lập hơn.



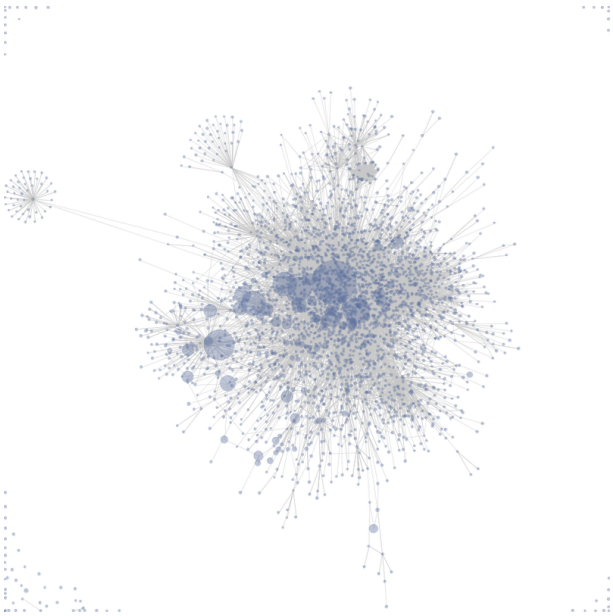
**Hình 2.15:** Circular layout

- **Spectral layout**
  Spectral layout sử dụng các eigenvectors của các ma trận, như là ma trận Laplace của đồ thị, làm tọa độ của các đỉnh. Bố cục này đặt mỗi đỉnh ở tâm của các đỉnh lân cận với một số sai lệch được xác định rõ ràng. Hai ưu điểm quan trọng của spectral layout là khả năng tính toán bố cục tối ưu (dựa theo yêu cầu cụ thể) và thời gian tính toán rất nhanh.



**Hình 2.16:** Spectral layout

- **Force-based layout**
  Force-based layout bố trí vị trí các đỉnh bằng cách liên tục di chuyển các đỉnh theo hệ thống các lực dựa trên các khái niệm vật lý. Thông thường, các hệ thống này kết hợp lực hấp dẫn giữa các đỉnh liền kề nhau với lực đẩy giữa tất cả các đỉnh. Ưu điểm của bố cục này bao gồm tính thẩm mỹ cao (chiều dài cạnh đều nhau, các đỉnh phân tách rõ ràng, thể hiện được tính đối xứng), linh hoạt, dễ dự đoán và dễ hiểu. Tuy vậy thời gian tính toán cho bố cục này lớn và chất lượng đồ thị sẽ phụ thuộc vào bố cục ngẫu nhiên ban đầu.



**Hình 2.17:** Force-based layout

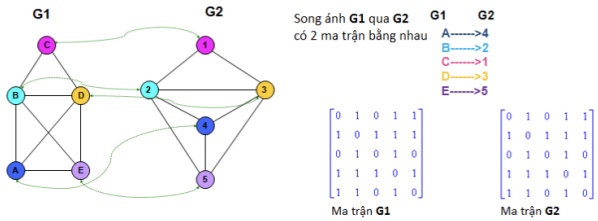
## 2.4 | Đồ thị đẳng cấu và đồ thị con đẳng cấu

### 2.4.1 | Đồ thị đẳng cấu

Hai đồ thị $G_1$ và $G_2$ được gọi là đẳng cấu với nhau, ký hiệu là $G_1 \approx G_2$, nếu tồn tại một hàm song ánh f:$V(G_1) \to V(G_2)$ với tính chất: 2 đỉnh a, b liền kề nhau trong $G_1$ khi và chỉ khi f(a), f(b) liền kề nhau trong $G_2$ với tất cả cặp đỉnh a,b của $G_1$, hay nói cách khác tồn tại một song ánh giữa 2 tập đỉnh sao cho mối quan hệ liền kề được bảo toàn.
**Hai đồ thị đẳng cấu thì ta có**

- Cùng số đỉnh.
- Cùng số đỉnh bậc k, với mọi k nguyên dương $\geq 0$.
- Cùng số cạnh.
- Cùng số thành phần.

$\Longrightarrow$ Nếu hai đồ thị có ma trận kề (theo một thứ tự đỉnh nào đó) bằng nhau thì chúng đẳng cấu với nhau.



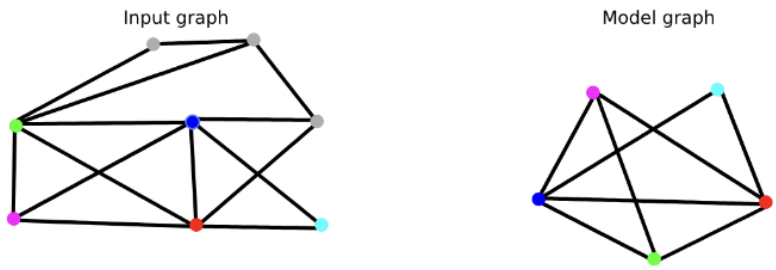

**Hình 2.18:** Đồ thị đẳng cấu

### 2.4.2 | Đẳng cấu đồ thị con

**Đồ thị con**
Đồ thị con H=(V',E') của đồ thị G=(V,E) được định nghĩa như sau:

- $V' \subseteq V$
- $E' \subseteq E$

Kí hiệu $H \subseteq G$ được dùng để biểu thị H là đồ thị con của G.
**Đẳng cấu đồ thị con**
Đồ thị G=(V,E) là đồ thị con đẳng cấu của đồ thị G'=(V',E') nếu $\exists$ S sao cho $S \subseteq G'$ và $S \approx G$. Khi đó, đồ thị G' được gọi là đồ thị đầu vào và đồ thị G là độ thị mẫu (model graph).
Bài toán đẳng cấu đồ thị con cho trước hai đồ thị G1 và G2, ta cần xác định xem G1 có chứa đồ thị con đẳng cấu với đồ thị G2 hay không. Bài toán được tổng quát hóa từ của hai bài toán cụ thể, một là bài toán tìm clique tối đa, hai là bài toán kiểm tra đồ thị có chứa chu trình Hamilton, và do đó nó là bài toán NP-complete. Tuy nhiên, trong một số trường hợp cụ thể bài toán đồ thị con đẳng cấu có thể được giải quyết trong thời gian đa thức.
**Ví dụ**: Trong ví dụ này các cặp đỉnh tương ứng với nhau thông qua phép song ánh đẳng cấu được đánh dấu cùng màu.



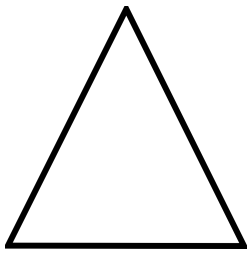

**Hình 2.19:** Đồ thị con đẳng cấu

## 2.5 | Tam giác, k-truss, và ý nghĩa của chúng đối với đồ thị bạn bè

### 2.5.1 | Tam giác

**Khái niệm**

Trong lý thuyết đồ thị, một tam giác là một tập hợp gồm ba đỉnh, trong đó mỗi đỉnh có mối quan hệ với tất cả các đỉnh khác



**Hình 2.20:** Tam giác.

**Ý nghĩa của tam giác trong việc phân tích đồ thị bạn bè:**

Trong mạng lưới xã hội, só tam giác khổng lồ thúc đẩy các nhà khoa học phát minh ra các chỉ số đo lường để mô tả và phân tích đồ thị. Các ứng dụng của việc đếm tam giác có thể kể đến tính toán hệ số phân cụm (cục bộ và toàn cục), tính bắc cầu và phát hiện các cộng đồng.

Một tính chất quan trọng của mạng lưới xã hội được tính toán dựa trên việc đếm tam giác là tính bắc cầu (transitivity) của mối quan hệ liền kề. Nếu mối quan hệ liền kề có tính bắc cầu, thì khi u liền kề với v, v liền kề với w thì u liền kề với w, hay diễn giải theo cách thông thường "bạn của bạn mình cũng là bạn mình". Tính bắc cầu một phần (partial transivity), thể hiện bằng hệ số phân cụm (clustering coefficient) được tính toán thông qua số tam giác trong đồ thị, được ứng dụng để tính xác suất để hai người có một người bạn chung là bạn của nhau.

$$C = \frac{3 \times \textit{só tam giác}}{\textit{số bộ ba}}$$

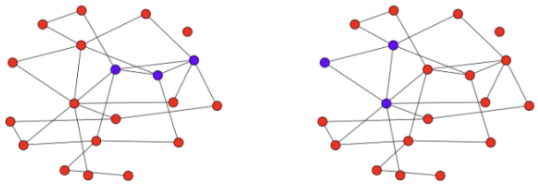
### 2.5.2 | K-truss

**Khái niệm**

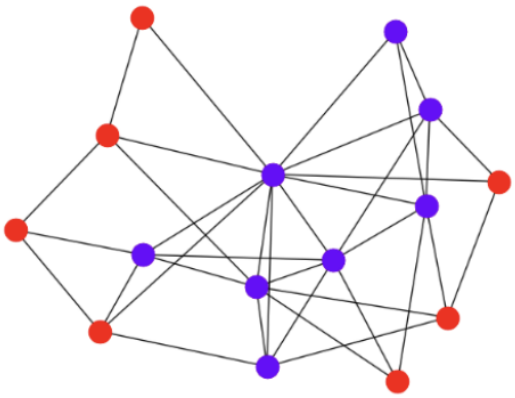
K–truss trong đồ thị là một đồ thị con mà mỗi cạnh của tập con đó thuộc ít nhất (k - 2) tam giác được tạo từ các điểm của truss. Một k-truss được gọi là tối đa nếu nó không phải là đồ thị con thực sự của một k-truss khác.

**Ví dụ**

- Trong đồ thị dưới đây ta có thể thấy 2 3-trusses trong đó. Theo định nghĩa, 3-truss là tam giác của đồ thị.



- Tương tự, đồ thị dưới đây có thể nhận thấy 4-truss của đồ thị. 4-trusses là tập hợp sao cho mỗi cạnh là một phần của 2 tam giác được tạo từ các điểm trong truss.



**Ý nghĩa của Ktruss trong việc phân tích đồ thị bạn bè:**

- Phân Loại Cộng Đồng và Mối Quan Hệ:
  - K-Truss có thể phân loại cộng đồng trong mạng xã hội bằng cách xem xét mức độ tương tác mạnh mẽ giữa các thành viên.
  - Các nhóm K-Truss đặc trưng cho sự tương tác chặt chẽ và đồng đều, cho phép xác định rõ ràng các mối quan hệ trong nhóm.
- Phân Tích Sự Lan Truyền Thông Tin:
  - Bằng cách phân rã mạng thành các nhóm K-Truss, chúng ta có thể phân tích cách thông tin lan truyền trong từng nhóm. Sự tương tác mạnh mẽ trong nhóm có thể ảnh hưởng đến việc thông tin được chia sẻ và lan truyền như thế nào.
  - Các nhóm K-Truss có thể chứa các "điểm nút" quan trọng trong việc phổ biến thông điệp hoặc ảnh hưởng đến quá trình lan truyền thông tin.
- Ứng Dụng Trong Chiến Dịch Quảng Cáo và Tiếp Thị:
  - Các nhóm K-Truss có thể là đối tượng quảng cáo hiệu quả vì sự tương tác mạnh mẽ trong nhóm giúp thông điệp quảng cáo được chuyển đạt và lan truyền hiệu quả hơn.

- Việc tập trung chiến dịch vào các nhóm này có thể giúp tiết kiệm nguồn lực và tăng hiệu suất chung.

- Dự Đoán Tương Tác Tương Lai:
  - Khi một nhóm được xác định là K-Truss, có thể dự đoán rằng mức độ tương tác giữa các thành viên sẽ tiếp tục trong tương lai. Điều này cung cấp thông tin quan trọng cho việc xây dựng chiến lược tương tác xã hội và duy trì mối quan hệ dài hạn.
- Tích Hợp Với Chiến Lược Xã Hội:
  - Hiểu rõ về cộng đồng K-Truss có thể giúp tích hợp nó vào chiến lược xã hội của tổ chức. Việc tập trung vào các nhóm có mối quan hệ mạnh mẽ có thể giúp tối ưu hóa chiến lược truyền thông và quảng cáo.

## 2.6 | Ý nghĩa bài toán đếm tam giác và phân rã truss đối với bài toán đồ thị đẳng cấu

2018 Static Graph Challenge tập trung vào các thuật toán tìm tam giác và k-trusses. Việc đếm tam giác có thể được coi là một trường hợp đặc biệt của bài toán đẳng cấu đồ thị con với tam giác là đồ thị con. Các thuật toán tìm hình tam giác không tìm thấy các đồ thị con trùng khớp chung nhưng có thể là một phần của các thuật toán tìm được. Các thuật toán tìm k-trusses có thể được sử dụng để mô tả đặc tính của đồ thị (con). Nếu đặc điểm k-truss của hai đồ thị (phụ) không nhất quán thì điều này chứng tỏ rằng các đồ thị (con) này không đẳng cấu. Đặc điểm k-truss nhất quán chỉ ra rằng cần phải kiểm tra bổ sung để xác định xem có tồn tại đẳng cấu hay không. Do đó, việc cải thiện hiệu quả của thuật toán đếm tam giác và k-truss có thể dẫn đến cải thiện hiệu quả của thuật toán đẳng cấu đồ thị con.

# 3 | Hướng tiếp cận

## 3.1 | Đếm tam giác

Các hướng tiếp cận chính cho bài toán đếm tam giác trong đồ thị có thể đến:

- **Tính toán dựa trên việc nhân ma trận**
  - Ưu điểm:
    - Có thể được dùng để đếm số tam giác cho mỗi đỉnh.
    - Thuật toán làm việc với ma trận tương đối dễ hiểu và dễ hiện thực.
  - Nhược điểm:
    - Không phát hiện tam giác cụ thể, chỉ đếm tổng số tam giác.
    - Tốn dữ liệu lớn để lưu trữ và tính toán các ma trận nếu đồ thị có kích thước lớn.
- **Đếm và liệt kê tam giác**
  - Ưu điểm:
    - Thực hiện dựa trên việc duyệt qua các đường đi có độ dài 2 và kiểm tra xem có tạo tam giác không.
    - Liệt kê được danh sách các tam giác cụ thể.
  - Nhược điểm:
    - Một số thuật toán yêu cầu sắp xếp dữ liệu trước khi thực hiện.
    - Độ phức tạp của thuật toán còn lớn.
- **Đếm xấp xỉ tam giác (Approximate triangle counting)**
  - Ưu điểm:
    - Thuật toán xấp xỉ dựa trên việc lấy mẫu bộ ba (triple sampling based method) tỏ ra ưu việt về độ chính xác và thời gian thi hành .
    - Thời gian thi hành thuật toán nhỏ.
  - Nhược điểm:
    - Độ chính xác không tuyệt đối.

$\Longrightarrow$ Vì bài toán yêu cầu đếm số tam giác con trong đồ thị mà không cần phải xác định tam giác cụ thể và tệp dữ liệu không quá lớn nên thuật hướng tiếp cận đầu tiên đã được chọn để giải quyết bài toán.

Trong hướng tiếp cận đếm tam giác thông qua việc nhân ma trận, thuật toán xuất hiện sớm nhất được đề ra dựa trên tính toán $A^3$ với A là ma trận kề. Tuy nhiên với độ phức tạp lên đến $O(n^3)$, thuật toán tỏ ra kém hiệu quả khi làm việc với các đồ thị lớn. Để khắc phục, một số nhà nghiên cứu đã đề ra các thuật toán khác nhau như là Burkhardt, et. al; Alon, Yuster, and Zwick; Wolf, et. al.Trong số đó bài nghiên cứu đã lựa chọn thuật toán do Wolf, et.al đề ra vì thuật toán tương đối dễ hiểu, dễ hiện thực và số phép nhân ma trận ít.

**Thuật toán Wolf, et al.:**

Thuật toán do Wolf, et al. đưa ra dựa trên ý tưởng rằng mỗi hàng x trong ma trận kề ngầm định lưu trữ tất cả các đường đi có độ dài bằng 2 với điểm giữa là x. Bằng cách nhân những cặp này với các cột của ma trận liên thuộc, ta đang tìm các cạnh kết nối 2 điểm cuối của đường đi trên, và từ đó tạo thành tam giác.

**Đầu vào:** Ma trận kề A và ma trận liên thuộc E của đồ thị lớn G.

**Đầu ra:** Số lượng tam giác $n_T$ có trong đồ thị G.

**Quy trình:** Sử dụng phép nhân ma trận $C = A \times E$, trong đó $C_{ij} = 2 \Leftrightarrow \exists x, y \in V(A_{ix} = A_{iy} = 1 \wedge E_{xj} = E_{yz} = 1)$, nghĩa là sẽ tồn tại 1 cặp đỉnh x,y kề với đỉnh i trong ma trận kề A, và liên thuộc cạnh j trong ma trận liên thuộc E, hay là sẽ tồn tại tam giác chứa đỉnh i và cạnh j. Kết quả ta sẽ có một ma trận C kích thước $n \times m$ với các phần tử có giá trị lần lượt bằng 0,1,2. Trong đó số tam giác được biểu thị qua số số 2 trong ma trận.

Từ đó ta có thể tính số lượng tam giác $n_T$ bằng cách cho số lượng phần tử bằng 2 trong ma trận C (nb) chia cho 3 vì mỗi tam giác được đếm ba lần (một lần cho mỗi đỉnh). Ngoài ra, ta có thể tính số tam giác chứa mỗi đỉnh hoặc mỗi cạnh bằng tính tổng số phần tử bằng 2 trong từng hàng hoặc cột tương ứng

**Mã giả**

**Algorithm 1** Triangle counting

C=$A \times E$
nb=find(C==2)
$n_T = \frac{nnz(C)}{3}$
**return** $n_T$

## 3.2 | Phân rã truss

Phân rã truss là bài toán xác định truss tối đa trong đồ thị để hiểu rõ hơn về mối quan hệ và tính chất của các phần trong đồ thị. Để giải quyết bài toán các nhà nghiên cứu đã đưa ra nhiều hướng tiếp cận khác nhau như:

- **Thuật toán I/0-efficient**
  - Gồm hai hướng tiếp cận là bottom-up và top-down.
  - Ưu điểm
    - Giảm thiểu bộ nhớ lưu trữ và không gian tìm kiếm.
    - Hiệu quả với việc tính toán với các k lớn.
  - Nhược điểm
    - Các tính toán cục bộ được áp dụng xuyên suốt khiến cho độ chính xác toàn cục khó được đảm bảo.
- **Thuật toán In-Memory**
  - Ưu điểm
    - Có thể chuyển thành các phép toán đại số tuyến tính, dễ dàng cho việc hiện thực hóa thuật toán.
  - Nhược điểm
    - Tốn dữ liệu lớn để lưu trữ và tính toán các ma trận nếu đồ thị có kích thước lớn.
    - Một vài thuật toán yêu cầu sắp xếp các cạnh trước khi tính toán.

Trong bài nghiên cứu này, bởi vì kích thước đồ thị không quá lớn cũng như để dễ hiện thực hóa thuật toán,ta sẽ làm việc với thuật toán In-Memory sử dụng các phép toán đại số tuyến tình, hay là thuật toán Array-based implementation of k-Truss.

**Thuật toán:**
Array-based implementation of k-Truss là một phương pháp triển khai thuật toán phân rã k-Truss trên đồ thị.

- Phương pháp này sử dụng ma trận liên thuộc và ma trận kề để biểu diễn đồ thị và tính toán.
- Sử dụng tích ma trận để tính toán số tam giác chứa các cạnh trong đồ thị. Số tam giác này liên quan đến việc xem xét sự chồng chéo giữa các lân cận của các đỉnh liên thuộc cạnh.
- Mã nguồn được cung cấp dưới dạng pseudocode, giúp dễ dàng tùy chỉnh và triển khai trên nhiều ngôn ngữ lập trình như MATLAB hoặc Python.
- Sử dụng thuật toán để tìm các cạnh không thuộc k-Truss, và sau đó loại bỏ chúng từ ma trận liên thuộc và điều chỉnh lại ma trận liền kề và lặp lại đến khi tối ưu.

**Các bước của thuật toán Array-based implementation of k-Truss:**
**Khởi tạo:**

- Đầu vào gồm ma trận liên thuộc E và ma trận kề A.
- Tạo ma trận R bằng cách nhân ma trận E với ma trận A. Ma trận R có tính chất tương tự với ma trận xuất hiện trong thuật toán đếm tam giác của Wolf, et al. được trình bày ở phần 3.1.

**Quy trình:**

- Thực hiện vòng lặp cho đến khi không còn cạnh nào cần loại bỏ.
- Tạo ma trận nhị phân s bằng cách xác định các phần tử trong R bằng 2 (R == 2).
- Tìm các cạnh cần xóa (x) bằng cách kiểm tra tổng của mỗi cột của s và lọc ra những cột có tổng nhỏ hơn k - 2.
- Nếu x không rỗng, thực hiện các bước sau:
  - Xóa các cạnh trong x khỏi ma trận liên thuộc E.
  - Cập nhật lại ma trận kề A để phù hợp với ma trận E mới.
  - Cập nhật ma trận s và tìm các cạnh cần xóa tiếp theo.
- Lặp lại bước trên cho đến khi x trở thành rỗng.

Kết quả:
Ma trận E cuối cùng sau khi loại bỏ các cạnh không thuộc k-Truss là kết quả cuối cùng của thuật toán.
**Mã giả:**

**Algorithm 2** Array based implementation of k-Truss algorithm

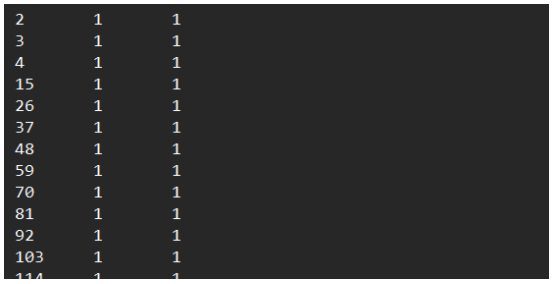
```
R=A × E
s=1 × (R == 2)
x=find(s<k-2)
while x is not empty do
    E_x = E(:, x)
    pos← nonzero indices each columm of E_x
    A(pos(:,0)pos(:,1)) = A(pos(:,1)pos(:,0)) = 0 {Cập nhật ma trận A}
    E=E(:,x_c) {x_c là phần bù của x trong tập cạnh của đồ thị}
    R=A × E
    s=1 × (R == 2)
    x=find(s<k-2)
end while
return E
```

## 3.3 | Xử lý dữ liệu

**Trong phạm vi assignment này, đầu vào đơn đồ thị vô hướng với cạnh có trọng số mặc định bằng 1 được cho với định dạng tập dữ liệu như sau:**

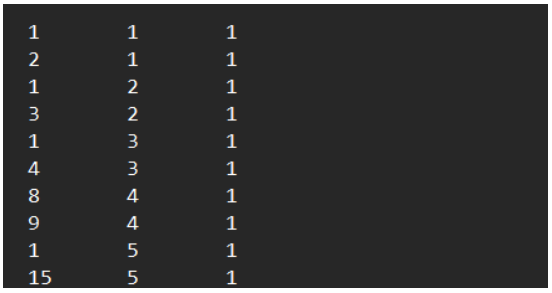
Tập dữ liệu gồm 2 tệp biểu diễn ma trận kề và ma trận liên thuộc được lưu dưới dạng tệp .tsv, tức là tệp dữ liệu phân tách bằng tab. Điều này có nghĩa là các giá trị trong mỗi hàng của tệp được phân tách bằng ký tự tab ($\backslash t$).

Trong từng tệp, số thứ tự của các đỉnh và cạnh là các số nguyên từ 1 đến số đỉnh n. Với dữ liệu biểu diễn ma trận kề, hai cột đầu mỗi hàng sẽ biểu diễn đỉnh bắt đầu và kết thúc của một cạnh và cột cuối sẽ biểu diễn trọng số của cạnh đó. Với dữ liệu biểu diễn ma trận liên thuộc, cột đầu biểu diễn số thứ tự của đỉnh liên thuộc với cạnh ở cột thứ hai, và cột cuối cũng sẽ biểu diễn trọng số của cạnh.



| | | |
|---|---|---|
| 2 | 1 | 1 |
| 3 | 1 | 1 |
| 4 | 1 | 1 |
| 15 | 1 | 1 |
| 26 | 1 | 1 |
| 37 | 1 | 1 |
| 48 | 1 | 1 |
| 59 | 1 | 1 |
| 70 | 1 | 1 |
| 81 | 1 | 1 |
| 92 | 1 | 1 |
| 103 | 1 | 1 |

(a) File adj



| | | |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 1 |
| 1 | 2 | 1 |
| 3 | 2 | 1 |
| 1 | 3 | 1 |
| 4 | 3 | 1 |
| 8 | 4 | 1 |
| 9 | 4 | 1 |
| 1 | 5 | 1 |
| 15 | 5 | 1 |

(b) File inc

Giá trị của các file sẽ được đọc vào chương trình và được xử lý để biến đổi thành ma trận kề và ma trận liên thuộc có kích thước lần lượt là $n \times n$ và $n \times m$, với n, m là số đỉnh và số cạnh của đồ thị.

Tuy nhiên, ta nhận thấy ma trận liên thuộc chỉ chứa $\frac{2}{m}$ phần tử khác không, hay ta có ma trận liên thuộc là một ma trận thưa. Cùng với việc nhân hai ma trận có kích thước như trên theo định nghĩa bao gồm nhân và cộng các phần tử sẽ tốn rất nhiều thời gian xử lý, ta sẽ lưu trữ ma trận liên thuộc dưới định dạng nén CSR (Compressed Sparse Row), một định dạng hiệu quả để lưu trữ và tính toán ma trận thưa thớt. [4]

## 3.4 | Trực quan hóa đồ thị

Để làm rõ hơn mối quan hệ giữa đồ thị con k-truss tối đa và đồ thị đầu vào G, nhóm quyết định sẽ trực quan hóa k-truss tối đa một cách độc lập và trong mối tương quan với đồ thị đầu vào.

Bởi đồ thị đầu vào trong bài toán này có số lượng cạnh và đỉnh rất lớn, cùng với nhu cầu của bài toán đặt ra là phát hiện các cộng đồng có mạng lưới liên hệ mạnh với nhau, ta chọn forced-based layout (cụ thể là thuật toán Fruchterman-Reingold force-directed) với ưu điểm tính thẩm mỹ cao và dễ phân tích để trực quan hóa. Ngoài ra, vì đồ thị đầu vào là đơn đồ thị vô hướng có trọng số các cạnh bằng 1, các label cũng được bỏ qua để tăng khả năng truyền tải thông tin (readability) của đồ thị. Màu và độ trong suốt của các cạnh trong đồ thị cũng được điều chỉnh để làm nổi bật vị trí của k-truss trong đồ thị đầu vào.

# 4 | Thực nghiệm

## 4.1 | Thực nghiệm kiểm thử tính đúng đắn của thuật toán trên dataset có số lượng nhỏ

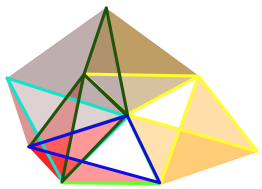
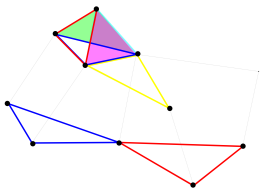
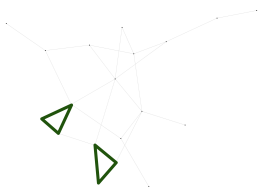
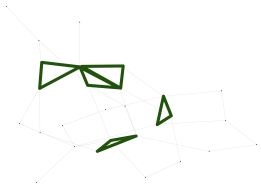
### 4.1.1 | Tổng quan dataset

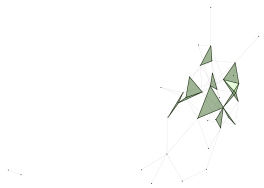
Để kiểm thử tính đúng đắn của thuật toán, một số dataset nhỏ với kích thước như sau được chạy thử.

| Dataset | Số đỉnh | Số cạnh |
|---|---|---|
| **1** | 9 | 22 |
| **2** | 11 | 20 |
| **3** | 19 | 27 |
| **4** | 28 | 43 |
| **5** | 38 | 75 |

### 4.1.2 | Thuật toán đếm tam giác

Kết quả chạy thử như sau:

| Dataset | Kết quả thuật toán | Kiểm thử |
|---|---|---|
| **1** | Number of triangle: 19.0 |  |
| **2** | Number of triangle: 7.0 |  |
| **3** | Number of triangle: 2.0 |  |
| **4** | Number of triangle: 5.0 |  |

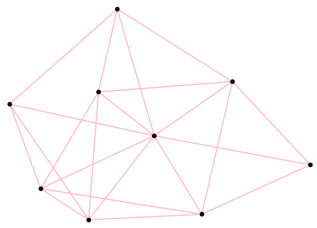
| Dataset | Kết quả thuật toán | Kiểm thử |
|---|---|---|
| 5 | Number of triangle: 12.0 |  |

$\Longrightarrow$ Qua quá trình kiểm thử ta có thể thấy được độ tin cậy của thuật toán đếm tam giác.

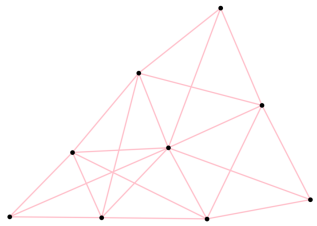
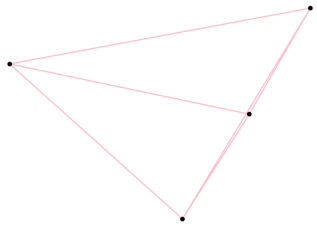
### 4.1.3 | Thuật toán phân rã k-truss

Kết quả chạy thử trên tập dataset 1,2 và 5 với các giá trị k như sau:

- k=3 (bao gồm tất cả tam giác trong đồ thị)

| Dataset | Kết quả thuật toán |
|---|---|
| 1 |  |
| 2 | |
| 5 | |

- k=4

| Dataset | Kết quả thuật toán |
|---|---|
| **1** |  |
| **2** |  |
| **5** | Không tồn tại 4-truss |

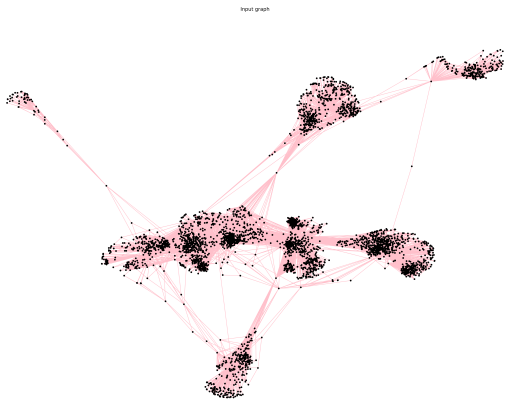
- Với các k lớn hơn, không dataset nào tồn tại truss phù hợp

$\Longrightarrow$ Qua quá trình kiểm thử ta có thể thấy được độ tin cậy của thuật toán phân rã k-truss.

## 4.2 | Áp dụng thuật toán trên dataset thực tế từ mạng xã hội Facebook

### 4.2.1 | Tổng quan dataset

Trong bài nghiên cứu này ta làm việc với tập dataset facebook_combined thực tế được lấy từ SNAP, nguồn cung cấp dữ liệu các network lớn của Stanford. Dataset này biểu diễn đồ thị cấu thành từ tập hợp các vòng tròn bạn bè trên Facebook, và được thu thập dữ liệu từ người dùng ẩn danh thông qua khảo sát. Đồ thị được cho gồm 4039 đỉnh và 88234 cạnh[2], là đơn đồ thị vô hướng với cạnh có trọng số mặc định bằng 1.




[2]Social circles: Facebook. Truy cập từ: https://snap.stanford.edu/data/ego-Facebook.html

### 4.2.2 | Bài toán đếm tam giác

Kết quả từ thuật toán:

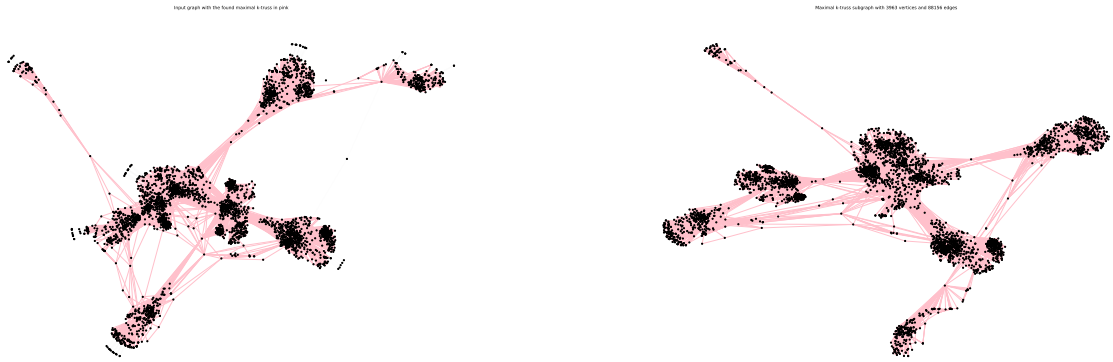
```
Number of triangle:
1612010.0
```

**Nhận xét:** Số tam giác trong tương quan với số đỉnh và cạnh tương đối lớn so với các đồ thị khác được cung cấp trong metadata của Graph challenge MIT [3], qua đó ta có thể thấy được tính bắc cầu của mạng lưới quan hệ bạn bè trên Facebook tương đối cao, nghĩa là khả năng cao 2 người bạn của người dùng bất kì cũng là bạn của nhau.

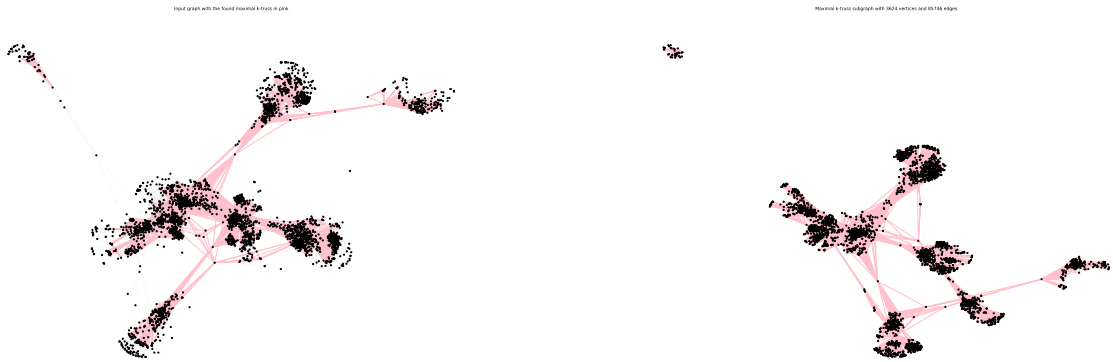
### 4.2.3 | Bài toán phân rã truss

Kết quả từ thuật toán với các k khác nhau:

- k=3



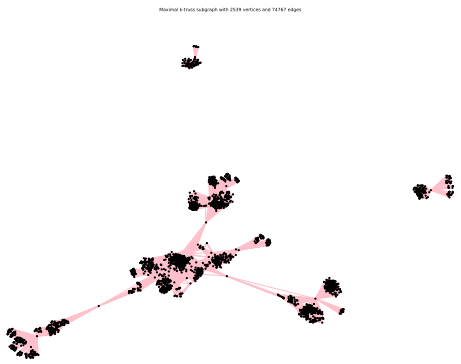
- k=5



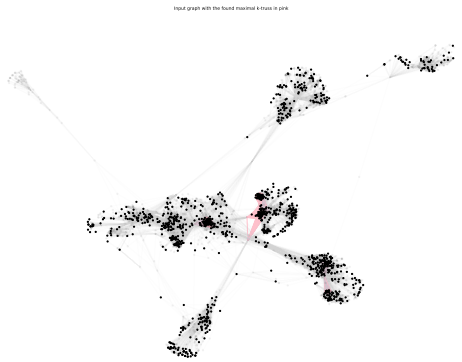
- k=10

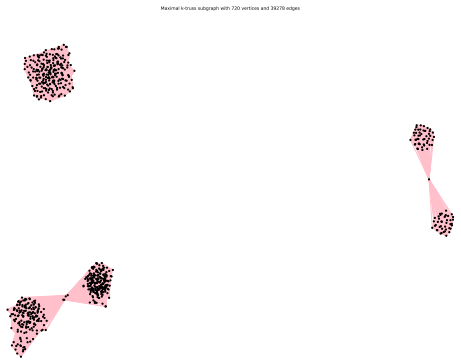
[3]Graph challeng MIT dataset. Truy cập từ: http://graphchallenge.mit.edu/data-sets#SNAPDatasets

Input graph with the found maximal k-truss in pink

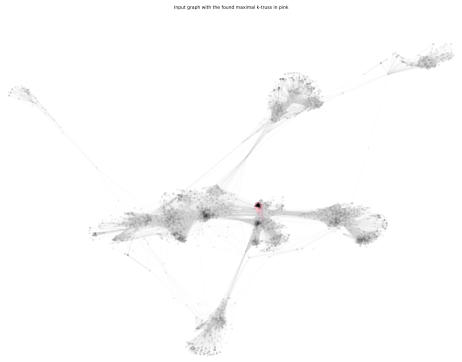

Maximal k-truss subgraph with 2539 vertices and 74747 edges

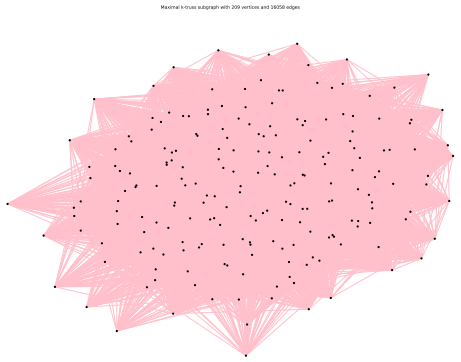
- k=30


Input graph with the found maximal k-truss in pink

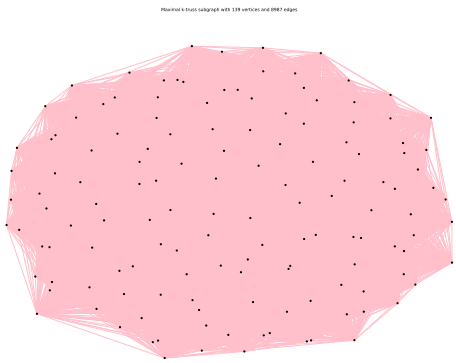

Maximal k-truss subgraph with 720 vertices and 39278 edges

- k=50

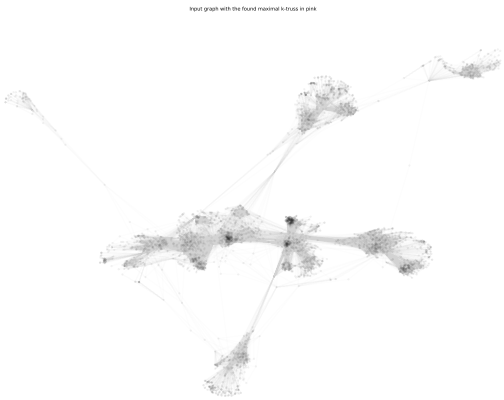

Input graph with the found maximal k-truss in pink


Maximal k-truss subgraph with 209 vertices and 16058 edges

- k=97




- k=98




**Nhận xét:**

Thông qua việc phân rã k-truss với các giá trị k khác nhau, ta phát hiện được các cộng đồng có mối liên hệ chặt chẽ tồn tại trong các cộng động lớn hơn với mối liên hệ ít chặt chẽ hơn.

Đồ thị tồn tại k-truss với k lớn nhất bằng 97, nghĩa là trong cộng đồng có mối quan hệ bền chặt nhất, mỗi mối quan hệ có ít nhất 95 bạn chung. Ngoài ra, gần 81% các mối quan hệ với 94% thành viên không thuộc 50-truss, vì vậy so với các cộng đồng khác thì cộng đồng 97-truss gồm 139 thành viên với 8987 mối quan hệ (3% người dùng và 10% mối quan hệ) có sự kết nối với nhau vượt bậc.

Đồng thời ta nhận thấy 50-truss và 97-truss tương đối giống nhau về hình dạng phân bố đều các đỉnh khi biểu diễn bằng force-based layout, đồng nghĩa với việc vị trí các đỉnh được quyết định bằng mối quan hệ liền kề của chúng, ta có thể đưa ra kết luận rằng trong cộng đồng 50-truss, các thành viên có mối quan hệ gắn khít tương đối đồng đều với tất cả thành viên còn lại. Trong khi đó, một số cộng đồng tồn tại trong 30-truss xuất hiện hiện tượng hai cộng đồng khác nhau được gắn kết bởi một nhóm người dùng nhất định có mối quan hệ chặt chẽ với cả hai.

98% người dùng và 99% tình bạn đều thuộc ít nhất 1 tam giác bạn bè, cho thấy đồ thị mối quan hệ bạn bè trên facebook có xác suất tồn tại bạn chung cao, phù hợp với kết quả đếm tam giác được nêu trên. 3-truss là một đồ thị liên thông và dựa trên kết quả thuật toán, chỉ từ 5-truss mới bắt đầu xuất hiện nhiều thành phần liên thông, qua đó ta thấy được sự liên kết gắn khít của tập người dùng facebook.

# 5 | Kết luận

Trong bài nghiên cứu này, ta đã tìm hiểu cơ sở lý thuyết của đồ thị và bài toán đồ thị con đẳng cấu, cách giải quyết bài toán trên với hai trường hợp xác định bao gồm phân rã k-truss và đếm tam giác. Cùng với đó ta đã giải thích được ý nghĩa của hai bài toán đó đối với đồ thị bạn bè của người dùng Facebook cũng như đối với bài toán đồ thị đẳng cấu. Cuối cùng, ta áp dụng thuật toán trên dataset thực tế của người dùng Facebook và rút ra được các đặc điểm của đồ thị đã cho.

Thuật toán được sử dụng trong bài nghiên cứu này áp dụng chủ yếu các phép tính toán đại số tuyến tính, chủ yếu là nhân ma trận,vì thế ngoài ưu điểm là dễ hiện thực thuật toán thì thời gian thực hiện của thuật toán khá lâu cũng như sẽ tốn nhiều dữ liệu cho việc tính toán và lưu trữ ma trận. Nhằm khắc phục điều này, những hướng đi sau được đề xuất cho các bài nghiên cứu sắp tới, bao gồm:

- Tối ưu hóa định dạng lưu trữ cho các đồ thị thưa.
- Sử dụng mô hình khác để phù hợp với các đồ thị mạng lưới xã hội thực tế với kích thước lớn.
- Tối ưu hóa việc trực quan hóa đồ thị để làm rõ hơn mối quan hệ giữa cá nhân với cá nhân và giữa cộng đồng với cộng đồng.

Tổng kết lại, thông qua bài tập lớn “Subgraph isomorphism: identigying a given subgraph in a larger graph” lần này, nhóm đã phối hợp, bàn bạc với nhau để tìm ra hướng đi hợp lí để hoàn thành nhiệm vụ được giao. Bên cạnh đó, nhóm cũng đã được mở mang tầm mắt, được trao cơ hội để có thể nghiên cứu sâu hơn về lĩnh vực phân tích và tính toán trên đồ thị, được tiếp xúc với các thuật toán, hướng tiếp cận khác nhau để từ đó tổng hợp lại và giải quyết bài toán đếm tam giác và phân rã truss cũng như nêu lên được ý nghĩa của bài toán trong thực tiễn.

# 6 | Tài liệu

[1] Iit kanpur seminar. https://www.cse.iitk.ac.in/users/dsrkg/cs245/html/seminar/isomorph.pdf.

[2] Stanford large network dataset collection. https://snap.stanford.edu/data/.

[3] Vijay Gadepally, Jake Bolewski, Dan Hook, Dylan Hutchison, Ben Miller, and Jeremy Kepner. Graphulo: Linear algebra graph kernels for nosql databases. 08 2015.

[4] Jianhua Gao, Weixing Ji, Zhaonian Tan, and Yueyan Zhao. A systematic survey of general sparse matrix-matrix multiplication. 02 2020.

[5] Helen Gibson, Joe Faith, and Paul Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information Visualization*, 12:324–357, 07 2012.

[6] Mohammad Hasan and Vachik Dave. Triangle counting in large networks: a review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8, 10 2017.

[7] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Yu. Querying k-truss community in large and dynamic graphs. 06 2014.

[8] Long Jin, Yang Chen, Tianyi Wang, Pan Hui, and Athanasios Vasilakos. Understanding user behavior in online social networks: A survey. ieee communications magazine, 51(9), 144-150. *Communications Magazine, IEEE*, 51:144–150, 09 2013.

[9] Yehuda Koren. Drawing graphs by eigenvectors: Theory and practice. *Computers Mathematics with Applications*, 49:1867–1888, 06 2005.

[10] Keith McNulty. *Handbook of Graphs and Networks in People Analytics: With Examples in R and Python.* Chapman and Hall/CRC, 2022.

[11] Mark Newman. *Networks: An Introduction 1st Edition.* Oxford University Press, 2010.

[12] Kenneth Rosen. *Discrete Mathematics and Its Applications 8th Edition.* McGraw Hill, 2018.

[13] Siddharth Samsi, Vijay Gadepally, Michael Hurley, Michael Jones, Edward Kao, Sanjeev Mohindra, Paul Monticciolo, Albert Reuther, Steven Smith, William Song, Diane Staheli, and Jeremy Kepner. Static graph challenge: Subgraph isomorphism. pages 1–6, 09 2017.

[14] Jia Wang and James Cheng. Truss decomposition in massive networks. *Proceedings of the VLDB Endowment*, 5, 05 2012.

[15] Michael Wolf, Jonathan Berry, and Dylan Stark. A task-based linear algebra building blocks approach for scalable graph analytics. pages 1–6, 09 2015.

[16] Chuanyu Zong, Pengcheng Gong, Xin Zhang, Tao Qiu, Anzhen Zhang, and Meng-xiang Wang. *Efficient Size-Constrained (k, d)-Truss Community Search*, pages 405–420. 11 2023.