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


# BÁO CÁO
# KHAI THÁC DỮ LIỆU ĐỒ THỊ
## Task 1 - 21KHDL

**Nhóm 7 - Sinh viên thực hiện:**

21127104 - Đoàn Ngọc Mai

21127129 - Lê Nguyễn Kiều Oanh

21127229 - Dương Trường Bình

21127616 - Lê Phước Quang Huy

**Giảng viên hướng dẫn:**

Lê Nhựt Nam

# Mục lục

# 1 Group Information

| MSSV | Họ và tên | Công việc được phân công | Mức độ hoàn thành |
|---|---|---|---|
| 21127104 | Đoàn Ngọc Mai | • Frequent Subgraph Mining | 100% |
| 21127129 | Lê Nguyễn Kiều Oanh | • Link Prediction. | 100% |
| 21127229 | Dương Trường Bình | • Random Walk For Node Embeddings: DeepWalk. | 100% |
| 21127616 | Lê Phước Quang Huy | • Random Walk For Node Embeddings: node2vec. | 100% |

# 2 Current Status

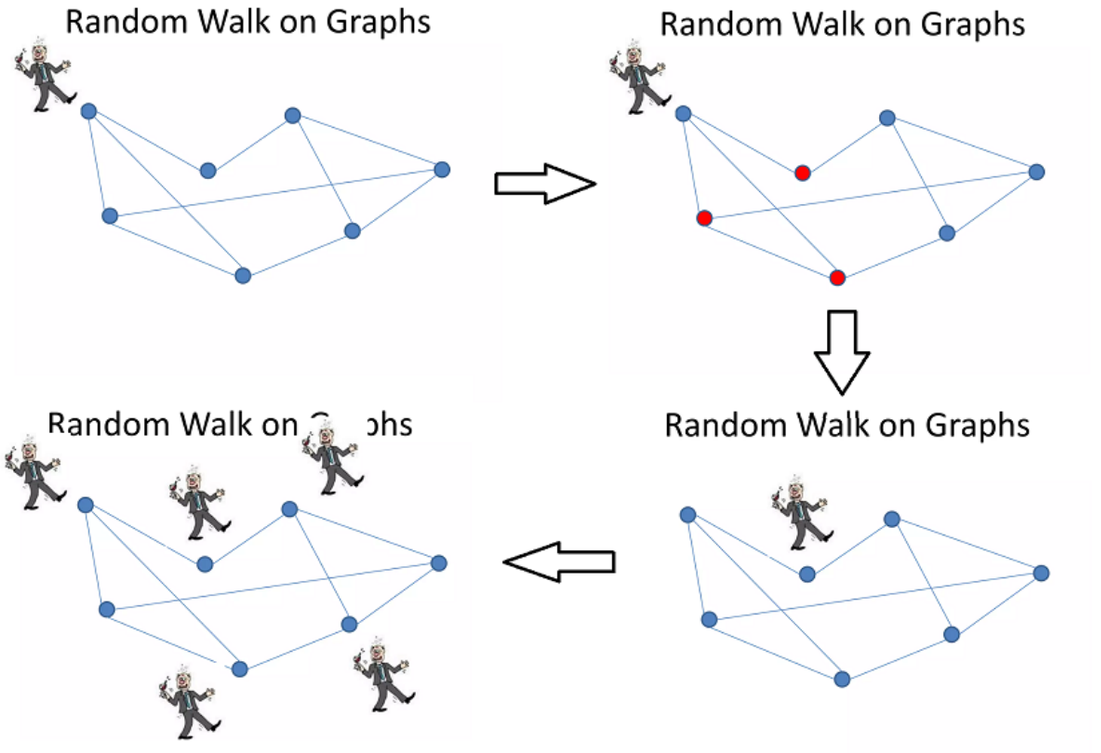
| Topic | Problem | Mức độ hoàn thành |
|---|---|---|
| Random Walk For Node Embeddings | - Giới Thiệu Tổng Quan | 100% |
| | - Các Khái Niệm Liên Quan Cần Biết | 100% |
| | - Quy trình chung | 100% |
| | - DeepWalk | 100% |
| | - Node2Vec 2 | 100% |
| Frequent Subgraph Mining | - Giới Thiệu Tổng Quan | 100% |
| | - Các Khái Niệm Liên Quan Cần Biết | 100% |
| | - Phương Pháp Tiếp Cận | 100% |
| | - Thách Thức Trong Frequent Subgraph Mining | 100% |
| | - Ứng Dụng Của Frequent Subgraph Mining | 100% |
| Link Prediction | - Giới Thiệu Tổng Quan | 100% |
| | - Kiến Thức Cơ Bản Cần Biết | 100% |
| | - Mô Hình Chung Cho Dự Đoán Liên Kết | 100% |
| | - Các phương pháp tiếp cận | 100% |

# 3 Techniques

## 3.1 Random Walk For Node Embeddings

**1. Giới Thiệu Tổng Quan**

- Trong bối cảnh đồ thị các thuật toán học máy truyền thống thường không làm việc hiệu quả với dữ liệu đồ thị do độ phức tạp cao của các mối quan hệ và cấu trúc trong đồ thị. Cần có các phương pháp đặc biệt để chuyển đổi các đỉnh trong đồ thị thành các vector số học dễ xử lý hơn. Thì Random Walk là một phương pháp phổ biến trong khai thác dữ liệu đồ thị, đặc biệt là trong việc tạo ra node embeddings.
- Với một đồ thị và một điểm bắt đầu, chúng ta chọn ngẫu nhiên (random) một lân cận (neighbor) của nó và di chuyển đến lân cận này; sau đó chúng ta chọn ngẫu nhiên một lân cận của điểm này và di chuyển đến lân cận đó, v.v. Chuỗi các điểm (ngẫu nhiên) được chọn theo cách này là một bước đi ngẫu nhiên trên đồ thị hay còn gọi là **Random Walk**.

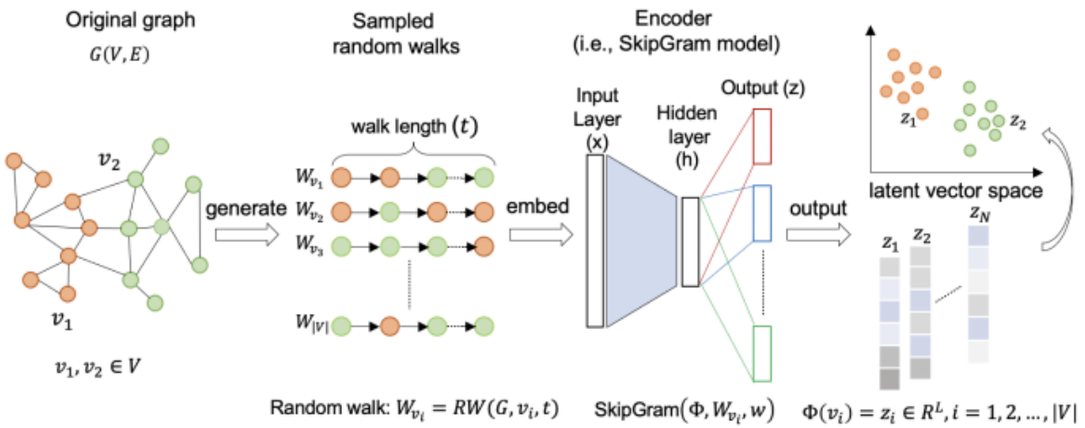



Hình 1: Random Walk trên đồ thị

**2. Các Khái Niệm Liên Quan Cần Biết**

- **Random Walk:** Như đã giải thích ở phần trên.
- **Node Embedding:** Biểu diễn vector của các đỉnh trong đồ thị trong một không gian vector có chiều thấp hơn. Node embeddings được học sao cho các đỉnh có liên kết hoặc mối quan hệ gần nhau trong đồ thị cũng có embeddings gần nhau trong không gian vector.
- **Context** : Trong ngữ cảnh của Node Embedding, "context"của một đỉnh là tập hợp các đỉnh khác mà nó có liên kết hoặc gần gũi với trong một chuỗi bước đi ngẫu nhiên. Skip-Gram sử dụng context này để học embeddings.
- **Skip-Gram:** Mô hình học sâu được sử dụng trong Word2Vec để dự đoán từ trong ngữ cảnh của nó. Trong đồ thị, Skip-Gram được áp dụng để học node embeddings từ các chuỗi bước đi ngẫu nhiên. Mục tiêu của Skip-Gram là tối đa hóa xác suất đồng thời xuất hiện của các đỉnh trong cùng một ngữ cảnh.
- **Negative Sampling:** Kỹ thuật lấy mẫu nhằm giảm chi phí tính toán trong quá trình tối ưu hóa mô hình Skip-Gram. Thay vì tối ưu hóa toàn bộ không gian đỉnh, Negative Sampling chỉ chọn một số lượng nhỏ các đỉnh không liên kết với nhau để tính toán, giúp giảm chi phí tính toán.
- **Biased Random Walk:** Một loại bước đi ngẫu nhiên được điều chỉnh bởi các tham số p và q để cân bằng giữa việc khám phá lân cận gần (BFS) và lân cận xa (DFS). Điều này cho phép Node2Vec kiểm soát hướng di chuyển trong quá trình bước đi ngẫu nhiên.
- **BFS (Tìm kiếm theo chiều rộng):** Một chiến lược tìm kiếm trong đó tất cả các đỉnh kề của một đỉnh được mở rộng trước khi di chuyển sâu hơn vào đồ thị. BFS tìm kiếm lân cận gần, do đó phù hợp với việc khám phá cộng đồng cục bộ trong đồ thị.
- **DFS (Tìm kiếm theo chiều sâu):** Một chiến lược tìm kiếm trong đó ta di chuyển sâu vào một nhánh trước khi quay lại mở rộng các đỉnh kề khác. DFS tìm kiếm lân cận xa, phù hợp với việc khám phá cấu trúc đồ thị toàn cục.

### 3. Quy trình chung




Hình 2: Pipeline for random walk-based graph embedding methods

Pipeline trên là quy trình tổng quát của phương pháp nhúng đồ thị dựa trên **random walk** để biểu diễn nút đồ thị. Dưới đây là giải thích chi tiết từng bước của quy trình:

- **Original Graph (Đồ thị gốc):**
  - Bắt đầu với một đồ thị (G = (V, E)), trong đó (V) là tập hợp các nút và (E) là tập hợp các cạnh.
  - Mục tiêu là tìm ra biểu diễn vector cho mỗi nút trong đồ thị này.
- **Sampled Random Walks (Mẫu bước đi ngẫu nhiên):**
  - Từ mỗi nút $v_i$ thực hiện nhiều bước đi ngẫu nhiên (random walks) với độ dài (t) để tạo ra một tập hợp các chuỗi các nút.
  - Công thức biểu diễn một random walk bắt đầu từ nút $v_i$ là $W_{v_i}$ = RW(G, $v_i$, t))
- **Encoder (Bộ mã hóa):**
  - Sau khi hoàn thành các bước đi ngẫu nhiên, chúng ta cần một bộ mã hóa (encoder) để biến đổi các chuỗi các nút (random walks) thành các embedding vectors.

  - Ở đây sử dụng mô hình SkipGram để mã hóa. SkipGram là một mô hình học sâu được sử dụng để dự đoán các nút lân cận từ một nút trung tâm trong random walk của nó.
  - Kết quả đầu ra từ mô hình SkipGram là các vector (z) biểu diễn cho các nút.

- **Latent Vector Space (Không gian vector tiềm ẩn):**
  - Các vector đầu ra $z_i$ sẽ được ánh xạ vào một không gian vector tiềm ẩn.
  - Những vector này là biểu diễn nhúng (embeddings) của các nút trong đồ thị, phản ánh cấu trúc của đồ thị và mối quan hệ giữa các nút.

**4. Phương Pháp Tiếp Cận**

Ở phần này chúng ta có 2 hướng tiếp cận là: **DeepWalk** và **Node2Vec**

- **DeepWalk:** Như hình mình họa ở trên đã trình bày quy trình tổng quát
  - **Chuẩn Bị Cho Chuỗi Random Walk:**
    - Chọn một đỉnh ngẫu nhiên từ đồ thị.
    - Di chuyển từ đỉnh hiện tại đến một đỉnh kề ngẫu nhiên. Lặp lại bước này để tạo ra một chuỗi các đỉnh. Độ dài thường được xác định trước và có thể điều chỉnh tùy thuộc vào yêu cầu và kích thước của đồ thị.
    - Thực hiện nhiều bước đi ngẫu nhiên từ nhiều đỉnh khác nhau để tạo ra một tập hợp các chuỗi đỉnh.
  - **Áp Dụng Mô Hình Skip-Gram:**
    - **Xác Định Context:** Trong mỗi chuỗi bước đi ngẫu nhiên, xác định các đỉnh lân cận xung quanh một đỉnh trung tâm. Context là tập hợp các đỉnh lân cận của đỉnh trung tâm trong chuỗi.
    - **Áp Dụng Mô Hình Skip-Gram:** Sử dụng mô hình Skip-Gram để học embeddings cho các đỉnh. Mô hình này cố gắng tối đa hóa xác suất của các đỉnh lân cận xuất hiện trong "context"của đỉnh trung tâm.
      - **Hàm mục tiêu:**

$$\max \sum_{u \in V} \sum_{v \in \mathscr{C}(u)} \log p(v|u)$$

        Trong đó:
      - V là tập các đỉnh trong đồ thị.
      - C(u) là tập các đỉnh trong ngữ cảnh của đỉnh u
      - p(v|u) là xác suất của v xuất hiện trong ngữ cảnh của u.

* **Negative Sampling:** Để giảm chi phí tính toán, DeepWalk sử dụng negative sampling, chọn một số đỉnh ngẫu nhiên không liên kết với đỉnh trung tâm và tối ưu hóa xác suất để chúng không xuất hiện trong ngữ cảnh của đỉnh trung tâm.

  · **Công Thức Negative Sampling:**

$$\log \sigma(\vec{v} \cdot \vec{u}) + \sum_{k=1}^{K} \mathbb{E}_{v_k \sim P_n(v)} \left[ \log \sigma(-\vec{v_k} \cdot \vec{u}) \right]$$

  Trong đó:

  · $\sigma(x)$ là hàm sigmoid.

  · $v_k$ là các đỉnh ngẫu nhiên được lấy mẫu từ phân phối nhiễu $P_n(v)$

  · K là số lượng đỉnh được lấy mẫu.

**– Ưu Điểm của DeepWalk:**

* Bằng cách sử dụng các bước đi ngẫu nhiên, DeepWalk khám phá cấu trúc đồ thị một cách hiệu quả, giúp học các embeddings chất lượng cao.

* Phương pháp này có thể áp dụng cho các đồ thị lớn nhờ vào việc sử dụng các chuỗi bước đi ngẫu nhiên và kỹ thuật negative sampling để giảm chi phí tính toán.

**– Hạn Chế của DeepWalk:**

* **Không Đảm Bảo Cấu Trúc Toàn Cục:** DeepWalk chủ yếu tập trung vào cấu trúc cục bộ (local) của đồ thị thông qua các bước đi ngẫu nhiên, có thể không nắm bắt được cấu trúc toàn cục của đồ thị một cách hoàn hảo.

* Các tham số như chiều dài chuỗi bước đi và số lượng bước đi có thể ảnh hưởng lớn đến chất lượng của embeddings, và việc chọn chúng là một thách thức.

- **Node2Vec:** Ở **DeepWalk** hạn chế của kỹ thuật đó là chủ yếu tập trung vào cấu trúc cục bộ. Thì **node2vec** sẽ khắc phục hạn chế đó bằng cách sử dụng bước đi ngẫu nhiên có điều chỉnh (biased random walk). Phương pháp này cố gắng cân bằng giữa việc khám phá cấu trúc cục bộ và cấu trúc toàn cục của đồ thị.

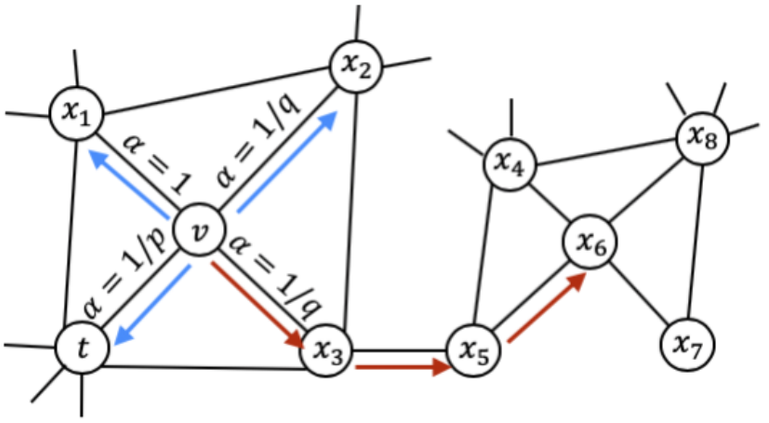
  – **Biased Random Walk:**

  * **Tham số p:** Điều chỉnh khả năng quay lại đỉnh trước đó. Nếu p cao, random walk ít có khả năng quay lại đỉnh trước đó, dẫn đến việc khám phá xa hơn trong đồ thị. Tương đương với việc thực hiện bước đi theo kiểu Depth-First Search (DFS). Ngược lại nếu p thấp thì tương đương với việc thực hiện bước đi theo kiểu Breadth-First Search (BFS).
  * **Tham số q:** Điều chỉnh khả năng di chuyển đến các đỉnh xa hơn trong đồ thị. Nếu q cao, bước đi ngẫu nhiên có khả năng di chuyển đến các đỉnh xa hơn thay vì chỉ ở trong khu vực gần gũi.
  * Xác suất chuyển tiếp từ đỉnh t đến đỉnh v, biết đỉnh trước đó là s:

$$\alpha_{pq}(t,v) = \begin{cases} \frac{1}{p} & \text{nếu } d_{tv} = 0 \text{ (quay lại đỉnh trước đó)} \\ 1 & \text{nếu } d_{tv} = 1 \text{ (di chuyển đến đỉnh lân cận trực tiếp)} \\ \frac{1}{q} & \text{nếu } d_{tv} = 2 \text{ (di chuyển xa hơn)} \end{cases}$$

  Trong đó:

  * $d_t v$ là khoảng cách ngắn nhất từ t đến v.




**Biased random walk schema incorporating both BFS and DFS.** Blue arrows show the breadth-first search (BFS) directions while red arrows indicate the depth-first search (DFS) directions.

Hình 3: Biased random walk schema incorporating both BFS and DFS

– **Tạo Chuỗi Random Walk:**

∗ **Khởi Tạo:** Chọn một đỉnh ngẫu nhiên từ đồ thị.

∗ **Thực Hiện Biased Random Walk:** Di chuyển từ đỉnh hiện tại đến các đỉnh kề, dựa trên xác suất chuyển tiếp điều chỉnh bởi tham số p và q. Lặp lại quá trình này để tạo ra chuỗi các đỉnh.

– **Áp Dụng Mô Hình Skip-Gram:**

∗ **Xác Định Context:** Tương tự như DeepWalk "context"là tập hợp các đỉnh lân cận của đỉnh trung tâm trong chuỗi.

∗ Sử dụng mô hình Skip-Gram để học embeddings cho các đỉnh từ các chuỗi bước đi ngẫu nhiên.

· **Hàm mục tiêu:**

$$\max \sum_{u \in V} \sum_{v \in \mathscr{C}(u)} \log p(v|u)$$

· **Negative Sampling:** Để giảm chi phí tính toán, Node2Vec sử dụng negative sampling tương tự như DeepWalk.

– **Ưu Điểm của Node2Vec:**

* **Cân Bằng Giữa Cấu Trúc Cục Bộ và Toàn Cục:** Bằng cách điều chỉnh tham số p và q, Node2Vec có thể cân bằng giữa việc khám phá cấu trúc cục bộ (BFS) và cấu trúc toàn cục (DFS) của đồ thị.
* **Linh Hoạt và Tinh Chỉnh:** Tham số điều chỉnh p và q cung cấp khả năng tùy chỉnh cao, giúp phù hợp với các loại đồ thị khác nhau và các mục đích học embeddings khác nhau.

– **Hạn Chế của Node2Vec:**

* **Chọn Tham Số:** Việc chọn giá trị cho các tham số p và q có thể yêu cầu thử nghiệm và tinh chỉnh để đạt được kết quả tốt nhất.
* **Chi Phí Tính Toán:** Mặc dù Node2Vec cải thiện hiệu quả bước đi ngẫu nhiên, việc tính toán embeddings cho đồ thị lớn vẫn có thể tiêu tốn nhiều tài nguyên.

**5. Ứng Dụng**

- Phát Hiện Cộng Đồng
- Khuyến Nghị Nội Dung
- Phân Tích Mạng Xã Hội Tội Phạm
- Cải Thiện Các Tính Năng Ngữ Nghĩa
- Hệ Thống Đề Xuất

## 3.2 Frequent Subgraph Mining

**1. Giới Thiệu Tổng Quan**

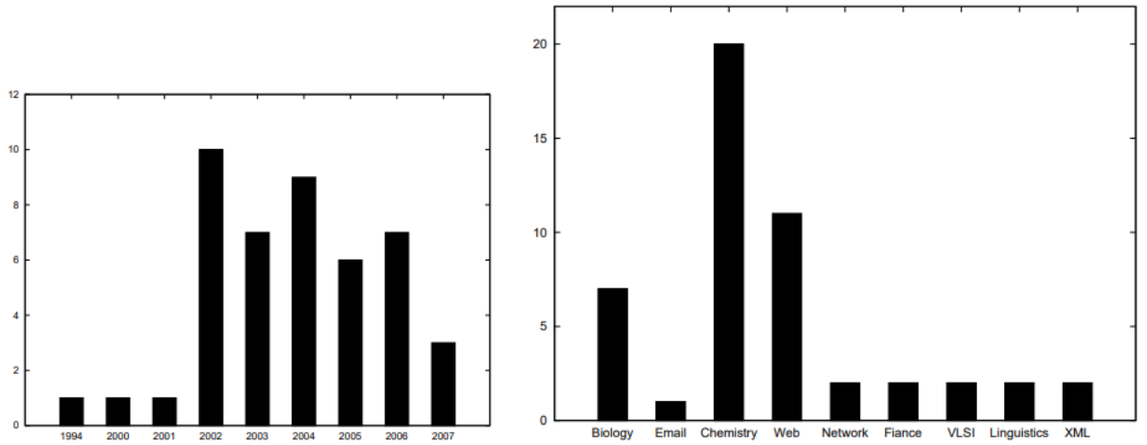
Frequent Subgraph Mining (FSM) là một kỹ thuật cốt lõi trong lĩnh vực khai phá đồ thị (graph mining). Mục tiêu của FSM là trích xuất tất cả các đồ thị con thường xuyên xuất hiện trong một tập dữ liệu đã cho, mà số lần xuất hiện của chúng vượt qua một ngưỡng nhất định. Kỹ thuật này có nhiều ứng dụng trong các lĩnh vực như hóa học, sinh học, và mạng xã hội.

| |
|---|
| Frequent subgraph mining (Cook & Holder 1994,2000; Inokuchi et al. 2000; Yan & Han 2002) |
| Correlated graph pattern mining (Ke et al. 2007; Ke et al. 2009; Ozaki & Ohkawa 2008) |
| Optimal graph pattern mining (Yan et al. 2008; Fan et al. 2008) |
| Approximate graph pattern mining (Kelley et al. 2003; Sharan et al. 2005; Chen et al. 2007a) |
| Graph pattern summarization (Xin et al. 2006; Chen et al. 2008) |
| Graph classification (Huan et al. 2004; Kudo et al. 2004; Deshpande et al. 2005) |
| Graph clustering (Flake et al. 2004; Huang & Lai 2006; Newman 2004) |
| Graph indexing (Shasha et al. 2002; Yan et al. 2004) |
| Graph searching (Yan et al. 2005b; Yan et al. 2006; Chen et al. 2007b) |
| Graph kernels (Gärtner et al. 2003; Kashima et al. 2003; Borgwardt & Kriegel 2005) |
| Link mining (Chakrabarti et al. 1999; Kosala & Blockeel 2000; Getoor & Diehl 2005; Liu 2008) |
| Web structure mining (kleinberg 1998; Brin & Page 1998) |
| Work-flow mining (Greco et al. 2005) |
| Biological network mining (Hu et al. 2005) |

Hình 4: Tổng quan về FSM theo số lượng các thuật toán được đề xuất và lĩnh vực ứng dụng.

Từ hình 4 có thể thấy các giai đoạn hoạt động trong đầu những năm 90 (trùng với việc giới thiệu khái niệm khai phá dữ liệu) tiếp theo là một giai đoạn hoạt động từ năm 2002 đến 2007. Không có thuật toán "mới" nào được giới thiệu trong những năm đó, cho thấy rằng lĩnh vực này đang dần đạt tới độ chín muồi, mặc dù đã có nhiều công trình tập trung vào các biến thể của các thuật toán hiện có.

Ngoài hoạt động nghiên cứu liên quan đến FSM, tầm quan trọng của FSM còn được phản ánh qua nhiều lĩnh vực ứng dụng của nó.




Hình 5: Tổng quan về lĩnh vực ứng dụng của FSM.

Hình 5 trình bày tổng quan về lĩnh vực ứng dụng của FSM theo số lượng các thuật toán FSM được báo cáo trong tài liệu và lĩnh vực ứng dụng cụ thể mà chúng được hướng tới. Từ hình có thể thấy ba lĩnh vực ứng dụng (hóa học, web, và sinh học) chiếm ưu thế trong việc sử dụng các thuật toán FSM.

#### 2. Các Khái Niệm Liên Quan Cần Biết về Frequent Subgraph Mining

Frequent Subgraph Mining (FSM) là một kỹ thuật được sử dụng để xác định các mẫu hoặc đồ thị con phổ biến trong một tập hợp các đồ thị. Dưới đây là các khái niệm chính liên quan đến Frequent Subgraph Mining:

##### Đồ Thị Con và Đẳng Cấu Đồ Thị

- **Đồ Thị Con:** Một đồ thị $G' = (V', E')$ là một đồ thị con của đồ thị $G = (V, E)$ nếu $V' \subseteq V$ và $E' \subseteq E$.
- **Đẳng Cấu Đồ Thị:** Hai đồ thị $G_1 = (V_1, E_1)$ và $G_2 = (V_2, E_2)$ được gọi là đẳng cấu nếu chúng có cấu trúc giống nhau, nghĩa là có một ánh xạ $f : V_1 \to V_2$ sao cho một cạnh $(u, v) \in E_1$ khi và chỉ khi $(f(u), f(v)) \in E_2$.

- **Tập Trợ:** Tập trợ của $G$ là $D_G = \{G_i | G \subseteq G_i, G_i \in D\}$ trong đó ký hiệu $G \subseteq G_i$ ám chỉ $G$ là đẳng cấu con của $G_i$.
- **Độ Trợ:** Độ trợ được tính $\sigma(G) = \frac{|D_G|}{|D|}$
- **Đồ Thị Con Bậc k (k-subgraph):** Đồ thị con bậc $k$ là đồ thị con với $k$ đỉnh hoặc $k$ cạnh.
- **Sinh Ứng Cử Viên:** Để xác định hai ứng cử viên để hợp nhất, chúng ta cần kiểm tra tính đẳng cấu đồ thị.
- **Cắt Tỉa Ứng Cử Viên:** Để kiểm tra tính chất đóng cửa xuống, chúng ta cần kiểm tra tính đẳng cấu đồ thị.
- **Đếm Tần Suất:** Kiểm tra tính đẳng cấu đồ thị để xác minh sự tồn tại của một đồ thị con thường xuyên.

#### Mẫu Đồ Thị Con Phổ Biến

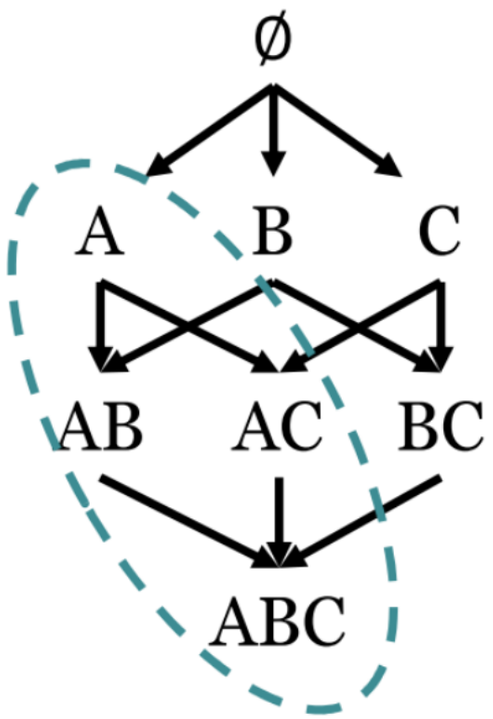
- **Mẫu Đồ Thị Con Phổ Biến:** Là một cấu trúc đồ thị con xuất hiện thường xuyên trong một tập các đồ thị cho trước. Một mẫu đồ thị con được xem là phổ biến nếu tần suất xuất hiện của nó lớn hơn hoặc bằng một ngưỡng tối thiểu (*minsup*).

#### Các Phương Pháp Khai Thác Mẫu Đồ Thị Phổ Biến

- **Phương Pháp Dựa Trên Apriori:** Sử dụng các k-subgraph phổ biến để phát sinh các (k+1)-subgraph phổ biến, dựa trên tính chất bao đóng hướng xuống (downward closure).
- **Phương Pháp Phát Triển Mẫu:** Mở rộng một đồ thị bằng cách thêm một cạnh mới và kiểm tra độ phổ biến của đồ thị mới này. Phương pháp này giúp giảm việc phát sinh ra các đồ thị trùng lặp và cải thiện việc kiểm tra đẳng cấu.
- **Phương Pháp Tham Lam (Greedy):** Tìm một số đồ thị con phổ biến nhất bằng cách sử dụng phương pháp tìm kiếm Beam Search, tiến hành theo từng mức và chỉ duyệt qua các nút tốt nhất tại mỗi cấp.

**3. Phương Pháp Tiếp Cận**

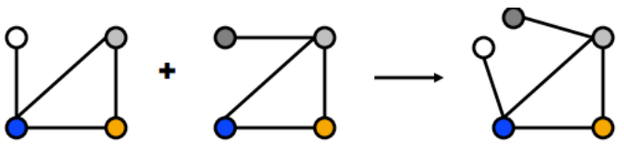
- **Phương Pháp Dựa Trên Apriori:**
  - Phương pháp này tương tự như việc khai thác tập hợp các mục thường xuyên và có tính đệ quy.
  - Ý tưởng của phương pháp dựa trên Apriori là sử dụng các $k$-subgraph phổ biến để phát sinh $k+1$ subgraph phổ biến.
  - Tính chất bao đóng hướng xuống (downward closure), nghĩa là nếu A không phổ biến thì các tập cha của nó cũng không phổ biến.




Hình 6: Minh họa tính chất bao đóng.

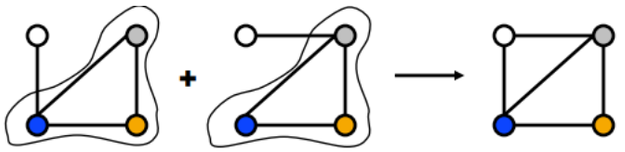
  - Một số thuật toán khai thác đồ thị con phổ biến dựa trên Apriori bao gồm:
    - **AGM:** Thuật toán này tạo ra các đồ thị ứng cử viên, hợp nhất bất kỳ hai đồ thị ứng cử viên tại một thời điểm và kiểm tra xem đồ thị kết quả có phải là một đồ thị con trong tập

đồ thị cho trước hay không. Kích thước của đồ thị được xác định bởi số lượng đỉnh có trong đồ thị đó. Hai đồ thị có kích thước '$k$ có thể được hợp nhất với nhau để tạo thành đồ thị kết quả có kích thước $k+1$.



Hình 7: Phát triển đỉnh (Vertex Growing) trong thuật toán AGM

* **FSG:** Thuật toán này sử dụng phương pháp sinh ứng cử viên dựa trên cạnh. Kích thước của một đồ thị được xác định bởi số lượng cạnh có trong đồ thị đó. Tương tự như phương pháp sinh ứng cử viên dựa trên đỉnh, hai đồ thị có kích thước $k$ được hợp nhất với nhau để tạo ra các đồ thị kết quả có kích thước $k+1$.



Hình 8: Phát triển đỉnh (Edge Growing) trong thuật toán FSG

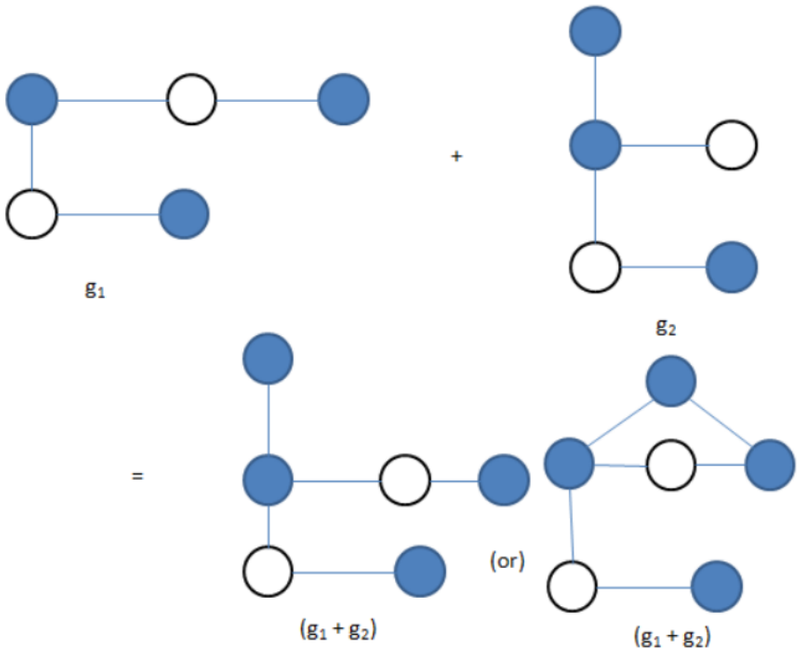
* **Thuật Toán Hợp Nhất Đường Đi Không Trùng Cạnh (Edge-disjoint path-join algorithm):** Thuật toán này dựa trên phương pháp tiếp cận Apriori và sử dụng các đường đi không trùng cạnh làm khối xây dựng. Yếu tố đo lường là số lượng các đường đi không trùng mà một đồ thị có được. Hai đồ thị ứng cử viên có $k$ đường đi không trùng được hợp nhất với nhau để tạo thành đồ thị kết quả có chứa $k+1$ đường đi không trùng.

– Hàm Phát Sinh Ứng Cử Viên Giữa AGM và FGM:

* AGM:

  · Hai đồ thị phổ biến kích thước $k$ được gia nhập chỉ nếu chúng có cùng đồ thị con kích thước $k-1$.

· Điểm khác so với Apriori khi làm trên đồ thị là việc gia nhập hai đồ thị có thể tạo ra nhiều hơn hai ứng viên.



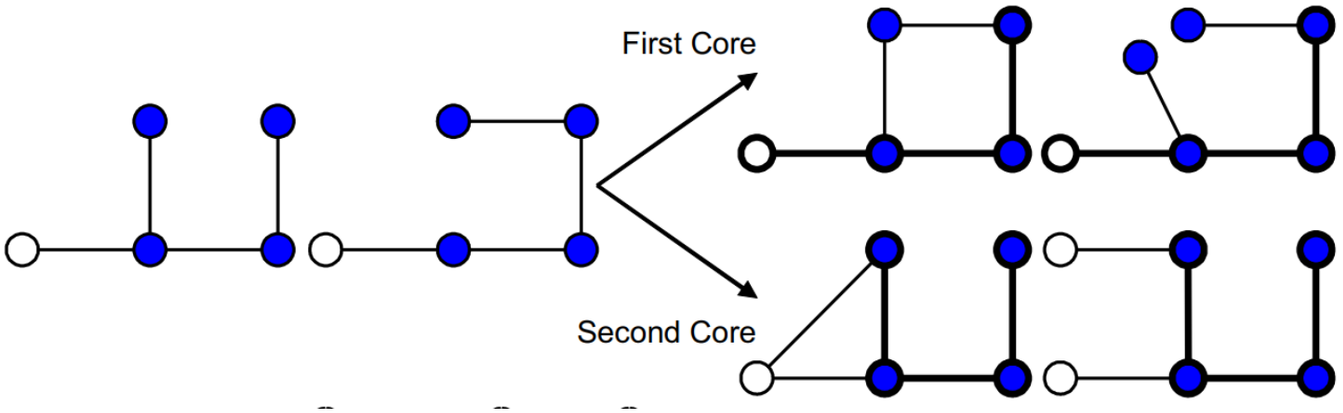

Hình 9: Đồ thị được gia nhập trong AGM

∗ FGM:

· Hai đồ thị phổ biến kích thước $k$ được gia nhập chỉ nếu chúng có cùng đồ thị con kích thước $k-1$ (đồ thị lõi - core).

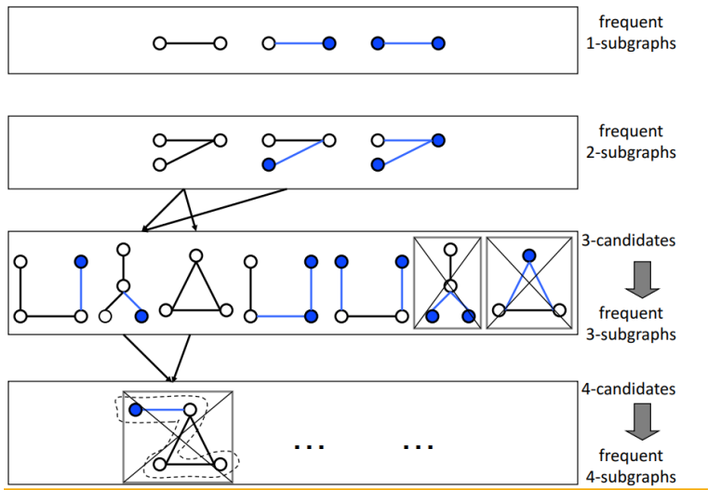
· **Cách xác định lõi (core) giữa hai đồ thị:** Tạo đồ thị con $(k-1)$ subgraph của đồ thị $G_i^k$ bằng cách bỏ đi một cạnh. Sau đó, tiến hành kiểm tra đồ thị con này có phải là đồ thị con của $G_j^k$ không. Sau cùng, hãy lặp lại quá trình trên để có những lõi khác (nếu có).

∗ Điểm khác so với Apriori khi làm trên đồ thị là việc gia nhập hai đồ thị có thể tạo ra nhiều hơn hai ứng viên.




Hình 10: Đồ thị được gia nhập trong FGM

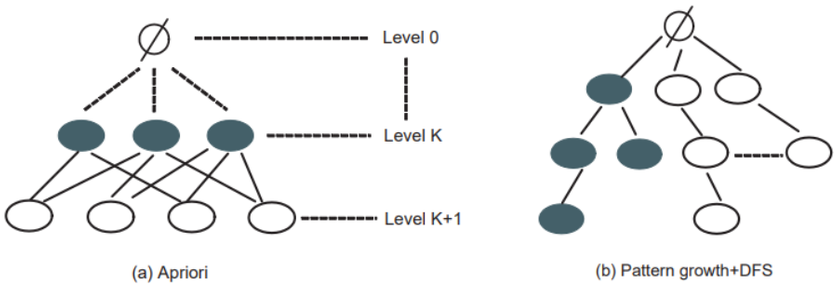
∗ Cắt Tỉa Ứng Cử Viên (tính chất bao đóng hướng xuống): Trong bước tỉa nhánh, mọi đồ thị con $(k-1)$ phải phổ biến. Đối với tất cả các đồ thị con $(k-1)$ của một ứng cử viên k bất kỳ, kiểm tra xem tính chất bao đóng hướng xuống có được thỏa mãn hay không.




Hình 11: Toàn bộ quá trình cắt tỉa trong FGM

- **Phương Pháp Phát Triển Mẫu:**
  - Phương pháp phát triển mẫu mở rộng một đồ thị con phổ biến bằng cách thêm một cạnh vào mọi vị trí có thể. Phương pháp này tránh được sự gia nhập hai đồ thị con có kích thước lớn để tạo thành đồ thị kích thước lớn hơn, nhưng có thể dẫn đến việc phát hiện lặp lại cùng

một đồ thị con nhiều lần.




Hình 12: Sự khác biệt giữa phương pháp Apriori và phương pháp mẫu

– Vấn đề trùng lặp này có thể được loại bỏ đáng kể bằng cách sử dụng kỹ thuật mở rộng theo đường đi bên phải (rightmost extension technique).

– Một số thuật toán sử dụng phương pháp phát triển mẫu bao gồm SPIN, MoFa, gSpan, FFSM, và Gaston.

– **MoFa (Molecular Fragments Identification Technique):**

  * Kỹ thuật này tìm các cấu trúc lõi (đồ thị con) có trong tất cả các cấu trúc phân tử được cho và tạo danh sách nhúng (embedding list).

  * Ở mức tiếp theo, mỗi cấu trúc trong danh sách nhúng được mở rộng bằng cách thêm một cạnh theo mọi cách có thể, tạo ra các cấu trúc khác nhau. Kỹ thuật cắt tỉa (pruning) được sử dụng để giảm bớt tính toán hỗ trợ và chỉ thực hiện mở rộng trên các cấu trúc trong danh sách nhúng.

  * Ở đây có thể phát sinh việc báo cáo lặp lại cùng một cấu trúc con, nhưng điều này có thể được loại bỏ bằng cách duy trì danh sách các cấu trúc con phổ biến và loại bỏ những cấu trúc mới trùng với những cấu trúc đã biết.

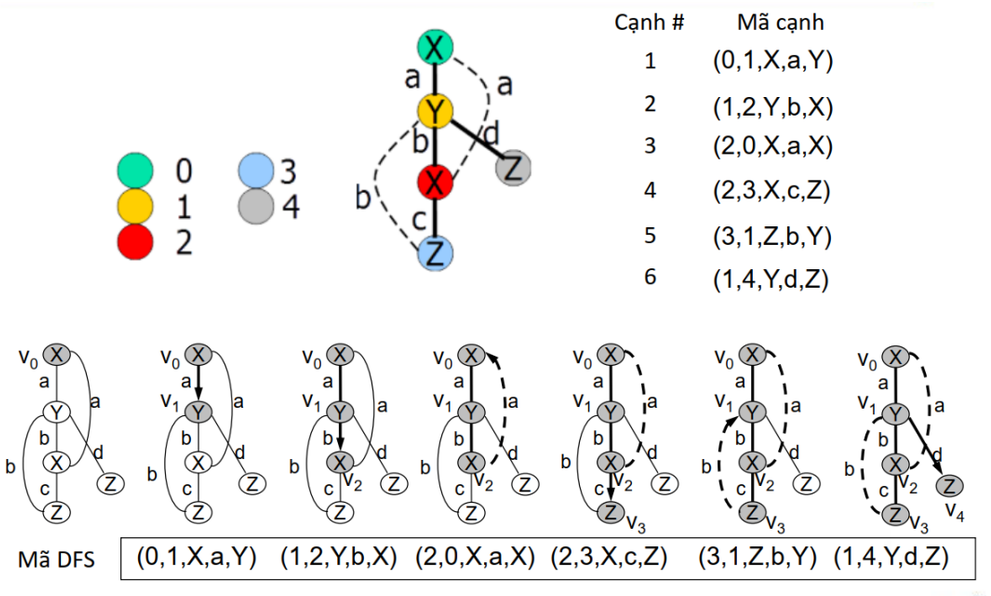
– **SPIN (Spanning Tree based Maximal Graph Mining):**

  * Thuật toán này khai thác các đồ thị con không thuộc về bất kỳ đồ thị con phổ biến nào khác.

* SPIN sử dụng phương pháp cây khung (spanning tree) để khám phá các đồ thị con phổ biến lớn nhất.

**– gSpan:**

* Thuật toán này tạo ra một cấu trúc dạng cây (DFS Code tree) chứa tất cả các mẫu đồ thị (subgraphs), trong đó mỗi nút đại diện cho một mã DFS (DFS Code) của một mẫu đồ thị.
* gSpan tránh được việc phát hiện trùng lặp các đồ thị bằng cách sử dụng kỹ thuật mở rộng theo đường đi bên phải.
* Thuật toán gSpan chuyển đồ thị thành tuần tự cạnh được mã hóa theo cách duyệt chiều sâu (DFS) và chọn mã DFS nhỏ nhất làm đại diện chính tắc cho đồ thị.
* Mã DFS là tuần tự của các cạnh được duyệt bằng DFS. Mỗi cạnh được mã hóa dưới dạng $(i, j, L_i, L_{(ij)}, L_j)$:
  - $i, j$: là các đỉnh.
  - $L_i, L_j$: nhãn đỉnh đầu và cuối của cạnh.
  - $L_{(ij)}$: nhãn cạnh.
  - $i < j$: được gọi là cạnh tới (forward edge).
  - $i > j$: được gọi là cạnh lùi (back edge).




Hình 13: Ví dụ minh họa về mã DFS

∗ **Luật Thứ Tự:**

· **Luật 1:** Nếu $i_1 = i_2$ và $j_1 < j_2$ thì $e_1 \prec e_2$.

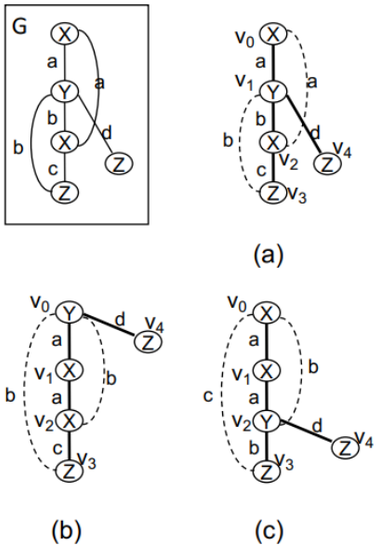
· Từ cùng một đỉnh nguồn, $e_1$ được duyệt trước $e_2$ trong DFS.

· **Luật 2:** Nếu $i_1 < j_1$ và $j_1 = i_2$ thì $e_1 \prec e_2$ (cạnh tới).

· $e_1$ là một cạnh tới và $e_2$ được duyệt như là kết quả của phép duyệt $e_1$.

· **Luật 3:** Nếu $e_1 \prec e_2$ và $e_2 \prec e_3$ thì $e_1 \prec e_3$ (bắc cầu).

| | (a) | (b) | (c) |
|---|---|---|---|
| 1 | (0, 1, X, a, Y) | (0, 1, Y, a, X) | (0, 1, X, a, X) |
| 2 | (1, 2, Y, b, X) | (1, 2, X, a, X) | (1, 2, X, a, Y) |
| 3 | (2, 0, X, a, X) | (2, 0, X, b, Y) | (2, 0, Y, b, X) |
| 4 | (2, 3, X, c, Z) | (2, 3, X, c, Z) | (2, 3, Y, b, Z) |
| 5 | (3, 1, Z, b, Y) | (3, 0, Z, b, Y) | (3, 0, Z, c, X) |
| 6 | (1, 4, Y, d, Z) | (0, 4, Y, d, Z) | (2, 4, Y, d, Z) |




Hình 14: Thứ tự cạnh hợp lệ trong mã DFS

**– Định Nghĩa** $a_t \prec_e b_t$**:**

* $a_t = (i_a, j_a, L_{i_a}, L_{i_a j_a}, L_{j_a})$

* $b_t = (i_b, j_b, L_{i_b}, L_{i_b j_b}, L_{j_b})$

* $a_t \prec_e b_t$ nếu:

a. Cả hai đều là cạnh tới (forward edges) và:

· $i_b < i_a$ (cạnh bắt đầu từ đỉnh được duyệt sau, tại sao?)

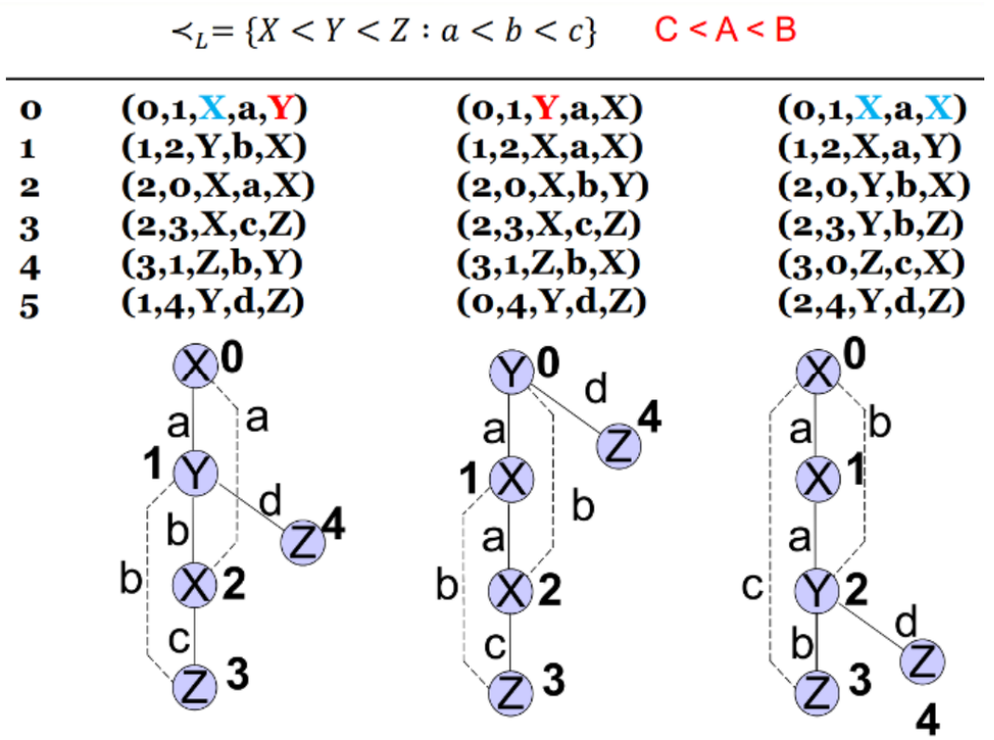
· $i_b = i_a$ và nhãn của $a$ nhỏ hơn nhãn của $b$ theo thứ tự từ điển.

b. Cả hai đều là cạnh lùi (backward edges) $(i_a = i_b)$ vì lý do tương tự và:

· $j_a < j_b$ (cạnh kết nối với đỉnh được khám phá trước đó).

· $j_a = j_b$ và nhãn cạnh của $a$ nhỏ hơn nhãn cạnh của $b$ theo thứ tự từ điển.

c. $a_t$ là cạnh lùi và $b_t$ là cạnh tới.




Hình 15: Ví dụ xét các mã DFS

  - gSpan cải thiện việc kiểm tra đẳng cấu bằng cách cắt bỏ các nhánh không cần thiết trong cây mã DFS nếu chúng không thỏa mãn điều kiện mã DFS nhỏ nhất.

- **Các Nghiên Cứu Khác:**
  - Ngoài các thuật toán khai thác đồ thị con phổ biến, còn có các phương pháp khai thác đồ thị con dựa trên ràng buộc (constraint-based subgraph mining).
  - Các nghiên cứu cũng được thực hiện để khai thác đồ thị con phổ biến trong các đồ thị lớn, với một số thuật toán nổi bật như HSIGRAM và VSIGRAM được đề xuất để khám phá tất cả các đồ thị con phổ biến.

**Phương Pháp Tìm Mẫu Phổ Biến Tham Lam:**

- Phương pháp tìm mẫu phổ biến tham lam sử dụng thuật toán Subdue, một thuật toán không đầy đủ nhằm tìm kiếm một số đồ thị con phổ biến nhất.
- Phương pháp này có thể không tìm được tất cả các đồ thị con phổ biến, nhưng có lợi thế là tốc

độ thực thi nhanh.

- Subdue dựa trên khái niệm chiều dài mô tả, tức là nén đồ thị bằng cách sử dụng các mẫu đồ thị để tối ưu hóa quá trình nén.
- Subdue sử dụng Beam Search, một dạng tìm kiếm giống BFS nhưng tiến triển qua từng mức chỉ bằng cách di chuyển qua W nút tốt nhất tại mỗi cấp. Các nút khác sẽ bị bỏ qua.

#### 4. Thách Thức Trong Frequent Subgraph Mining

- **Xác Định Đẳng Cấu Đồ Thị Con:** Là một bài toán NP-Complete, đòi hỏi phải kiểm tra tất cả các tổ hợp có thể có của các đỉnh để xác định tính đẳng cấu.
- **Giảm Không Gian Tìm Kiếm:** Các thuật toán hiệu quả phải giảm không gian tìm kiếm bằng cách giảm số lần kiểm tra đẳng cấu đồ thị con.

#### 5. Ứng Dụng Của Frequent Subgraph Mining

- **Tìm Con Đường Sinh Học:** Xác định các con đường sinh học chung giữa các loài.
- **Phân Tích Cộng Đồng Trong Mạng Xã Hội:** Tìm các nhóm hoặc cộng đồng có tính chất tương tự nhau trong mạng xã hội.
- **Xây Dựng Các Khối Cho Phân Lớp Đồ Thị:** Sử dụng các mẫu đồ thị phổ biến để phân lớp, gom nhóm, nén, so sánh hay phân tích tương quan.

## 3.3 Link Prediction

#### 1. Giới Thiệu Tổng Quan

Dự đoán liên kết (link prediction) là một bài toán quan trọng trong lĩnh vực khai phá mạng (network mining) và học máy (machine learning). Mục tiêu chính của dự đoán liên kết là xác định các liên kết tiềm năng giữa các nút trong một mạng mà hiện tại chưa tồn tại nhưng có khả năng cao sẽ hình thành trong tương lai. Kỹ thuật này có nhiều ứng dụng thực tế như:

- **Mạng Xã Hội (Social Networks):** Gợi ý kết bạn hoặc kết nối chuyên nghiệp trên các nền tảng như Facebook, LinkedIn.
- **Hệ Thống Khuyến Nghị (Recommendation Systems):** Gợi ý sản phẩm, dịch vụ trên các nền tảng thương mại điện tử như Amazon, Netflix.
- **Mạng Sinh Học (Biological Networks):** Dự đoán tương tác giữa các protein, giúp hiểu rõ hơn về các quá trình sinh học và bệnh lý.

#### 2. Kiến Thức Cơ Bản Cần Biết

Để hiểu và áp dụng các kỹ thuật dự đoán liên kết, cần nắm vững một số khái niệm cơ bản về mạng và các đặc trưng của chúng:

- **Mạng (Graph):** Một mạng (hoặc đồ thị) là một cấu trúc gồm các nút (đỉnh) và các cạnh (liên kết). Mạng có thể là có hướng hoặc vô hướng, có trọng số hoặc không trọng số.
- **Đỉnh (Node):** Đại diện cho các thực thể trong mạng (ví dụ: người dùng, sản phẩm, protein).
- **Liên kết (Edge):** Đại diện cho mối quan hệ giữa các đỉnh (ví dụ: quan hệ bạn bè, tương tác, giao dịch) hoặc kết nối giữa hai nút trong mạng
- **Đường Đi (Path):** Một chuỗi các cạnh kết nối một chuỗi các nút trong mạng.
- **Lân Cận (Neighborhood):** Tập hợp các nút kề (liên kết trực tiếp) với một nút cụ thể.
- **Tính Toán Độ Tương Đồng (Similarity Measures):** Được sử dụng để ước tính khả năng tồn

tại liên kết giữa hai nút. Các phương pháp phổ biến bao gồm tính toán độ tương đồng dựa trên lân cận hoặc dựa trên cấu trúc toàn mạng.

#### 3. Mô Hình Chung Cho Dự Đoán Liên Kết

Dự đoán liên kết là quá trình tính toán điểm số $s(u,v)$ cho mỗi cặp nút $\langle u,v\rangle$ trong một mạng, sau đó xếp hạng các cặp để chọn ra cặp có điểm số cao nhất để nối thêm một cạnh mới. Quá trình dự đoán liên kết bao gồm các bước chính sau:

- **Bước 1: Tính Toán Khoảng Cách Giữa Các Đỉnh**
  - Trong bước này, ta sử dụng một số phương pháp đo như Chỉ Số Jaccard, Đường Đi Ngắn Nhất, hoặc SimRank để tính toán khoảng cách giữa các cặp đỉnh trong mạng. Đây là giai đoạn quan trọng để xác định mức độ gần gũi giữa các đỉnh và khả năng kết nối của chúng.
- **Bước 2: Chọn Các Cặp Đỉnh Có Khoảng Cách Gần Nhất**
  - Dựa trên kết quả từ bước 1, ta chọn ra một số lượng cặp đỉnh có khoảng cách gần nhất (hoặc có điểm số tương đồng cao nhất) để tiếp tục phân tích và dự đoán liên kết.
- **Bước 3: Dự Đoán Cạnh Mới**
  - Từ danh sách các cặp đỉnh đã chọn ở bước 2, ta tiến hành dự đoán các cạnh mới có khả năng xuất hiện trong mạng. Các phương pháp dự đoán có thể bao gồm việc sử dụng các mô hình học máy hoặc các thuật toán đồ thị.
- **Bước 4: Đánh Giá Đồ Thị Được Dự Đoán với Đồ Thị Gốc**
  - Cuối cùng, ta so sánh đồ thị mới sau khi dự đoán liên kết với đồ thị gốc để đánh giá độ chính xác của các dự đoán. Các chỉ số đánh giá có thể bao gồm độ chính xác, độ nhạy, và F1-score.

#### 4. Các phương pháp tiếp cận

Các phương pháp tiếp cận được thảo luận dưới đây sẽ được sử dụng như những thước đo để tính toán khoảng cách giữa các đỉnh trong mạng lưới. Đây chính là bước đầu tiên trong mô hình chung đã được đề cập ở phần trước. Các phương pháp dự đoán liên kết thường được phân loại thành ba nhóm chính:

- **Phương Pháp Dựa Trên Đỉnh Láng Giềng (Neighborhood-Based Methods):** Các phương pháp trong nhóm này sử dụng thông tin từ các nút láng giềng (những nút kết nối trực tiếp với một nút cụ thể) để dự đoán khả năng tồn tại của một liên kết giữa hai nút. Dưới đây là các phương pháp tiêu biểu trong nhóm này:

  a. **Common Neighbors (Số Láng Giềng Chung)**

  – **Mô tả**: Đây là một trong những phương pháp đơn giản nhất. Giả định rằng nếu hai nút có nhiều láng giềng chung thì khả năng chúng sẽ kết nối với nhau là cao.

  – **Công thức**:

  $$CN(u,v) = |N(u) \cap N(v)|$$

  Trong đó, $N(u)$ và $N(v)$ là tập hợp các láng giềng của $u$ và $v$.

  – **Ví dụ**: Trong một mạng xã hội, nếu hai người bạn có nhiều bạn chung, khả năng cao họ sẽ kết bạn với nhau.

  – **Nhược Điểm:** Phương pháp này không tính đến sự khác biệt về số lượng láng giềng giữa các đỉnh, dẫn đến độ chính xác không cao trong nhiều trường hợp.

  b. **Jaccard Index (Hệ Số Jaccard)**

  – **Mô tả**: Hệ số Jaccard đo lường mức độ tương đồng giữa hai nút dựa trên tỷ lệ giữa số lượng láng giềng chung và tổng số láng giềng riêng biệt của hai nút đó. Phương pháp này giải quyết nhược điểm của Common Neighbors

– **Công thức**:

$$Jaccard(u,v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

– **Ví dụ**: Nếu $u$ có 5 láng giềng và $v$ có 7 láng giềng, trong đó có 3 láng giềng chung, chỉ số Jaccard sẽ là $\frac{3}{5+7-3} = \frac{3}{9} \approx 0.33$.

c. **Adamic Adar Coefficient (Độ Đo Adamic-Adar)**

– **Mô tả**: Độ đo này cải tiến từ phương pháp số láng giềng chung (Common Neighbors) bằng cách tăng trọng số cho các nút chung ít phổ biến hơn (nút có ít láng giềng).

– **Công thức**:

$$AdamicAdar(u,v) = \sum_{z \in N(u) \cap N(v)} \frac{1}{\log |N(z)|}$$

Trong đó, $z$ là một nút láng giềng chung giữa $u$ và $v$, và $|N(z)|$ là số láng giềng của nút $z$.

– **Ví dụ**: Nếu $z$ là một láng giềng chung giữa $u$ và $v$ và $z$ có rất ít láng giềng, giá trị của $\frac{1}{\log |N(z)|}$ sẽ cao, làm tăng giá trị Adamic-Adar.

- **Phương Pháp Dựa Trên Đường Đi (Path-Based Methods):** Sử dụng thông tin về các đường đi giữa các đỉnh để đưa ra dự đoán. Các phương pháp này thường sử dụng cấu trúc toàn mạng và các đường đi cụ thể để dự đoán các liên kết tiềm năng. Một số phương pháp phổ biến trong nhóm này bao gồm:

a. **Shortest Path (Đường Đi Ngắn Nhất)**

– **Mô tả**: Phương pháp này dựa trên độ dài của đường đi ngắn nhất giữa hai nút. Nếu có đường đi ngắn nhất thì khả năng tồn tại liên kết giữa hai nút là cao.

– **Công thức**: Đường đi ngắn nhất giữa hai nút $u$ và $v$ là số cạnh nhỏ nhất nối $u$ và $v$.

– **Ví dụ**: Trong một mạng xã hội, nếu hai người chỉ cần qua một người trung gian để kết nối, họ có khả năng cao sẽ trở thành bạn bè trực tiếp.

b. **PageRank**

- **Mô tả**: PageRank là một thuật toán xếp hạng các nút trong mạng dựa trên tầm quan trọng của chúng. Ý tưởng cơ bản là một nút càng được nhiều nút quan trọng khác liên kết đến thì nó càng quan trọng.
- **Công thức**:

$$PR(u) = \frac{1-d}{N} + d \sum_{v \in In(u)} \frac{PR(v)}{|Out(v)|}$$

Trong đó:

* $PR(u)$ là giá trị PageRank của nút $u$.
* $d$ là hệ số giảm dần (thường được chọn là 0.85).
* $N$ là tổng số nút trong mạng.
* $In(u)$ là tập hợp các nút liên kết đến $u$.
* $Out(v)$ là số liên kết đi ra từ nút $v$.

- **Ví dụ**: Trong mạng web, các trang có nhiều liên kết đến từ các trang có giá trị PageRank cao sẽ có giá trị PageRank cao hơn, và khả năng cao sẽ có thêm nhiều liên kết đến trong tương lai.

c. **SimRank**

- **Mô tả**: SimRank đo lường sự tương đồng giữa hai nút dựa trên nguyên tắc rằng hai nút càng giống nhau nếu các láng giềng của chúng cũng giống nhau.
- **Công thức**:

$$SimRank(u,v) = \frac{C}{|In(u)||In(v)|} \sum_{i \in In(u)} \sum_{j \in In(v)} SimRank(i,j)$$

Trong đó, $C$ là hệ số điều chỉnh (thường là 0.8), và $In(u)$ là tập hợp các nút liên kết đến $u$.

– **Ví dụ**: Trong một mạng sản phẩm, nếu hai sản phẩm thường xuyên được mua cùng nhau bởi những khách hàng tương tự, chúng sẽ có giá trị SimRank cao và có khả năng được kết nối lại với nhau trong một hệ thống khuyến nghị.

d. **Ensemble of Short Paths**

– **Mô tả**: Phương pháp "Ensemble of Short Paths"kết hợp thông tin từ nhiều đường đi ngắn nhất giữa hai nút để dự đoán khả năng liên kết. Thay vì chỉ sử dụng một đường đi ngắn nhất, phương pháp này tính toán điểm số cho một cặp nút dựa trên tập hợp tất cả các đường đi ngắn giữa chúng. Điều này giúp cải thiện độ chính xác dự đoán bằng cách xem xét nhiều kịch bản kết nối tiềm năng.

– **Công thức**:

$$Score(u,v) = \frac{1}{\sum_{p \in Paths(u,v)} Length(p)}$$

Trong đó, $Paths(u,v)$ là tập hợp tất cả các đường đi ngắn nhất giữa $u$ và $v$, và $Length(p)$ là độ dài của đường đi $p$.

– **Ví dụ**: Trong một mạng giao thông, việc tính toán điểm số cho hai địa điểm dựa trên tất cả các tuyến đường ngắn nhất giữa chúng có thể giúp dự đoán xem liệu chúng có thể kết nối trực tiếp trong tương lai hay không.

e. **Katz measure (Độ Đo Katz)**

– **Mô tả**: Độ đo Katz đo lường khả năng kết nối giữa hai nút dựa trên tổng số tất cả các đường đi từ một nút đến nút còn lại, được điều chỉnh bởi một hệ số giảm dần. Độ đo này có trọng số các đường đi ngắn hơn nhiều hơn các đường đi dài hơn, giúp phản ánh khả năng kết nối gần gũi hơn giữa các nút.

– **Công thức**:

$$Score(u,v) = \sum_{k=1}^{\infty} \beta^k \cdot \left(A^k\right)_{uv}$$

Trong đó:

* $\beta$ là hệ số giảm dần (thường $\beta < 1$) để điều chỉnh trọng số các đường đi.
* $A$ là ma trận kề của mạng.
* $\left(A^k\right)_{uv}$ là số lượng các đường đi dài $k$ giữa $u$ và $v$.

– **Ví dụ**: Trong một mạng xã hội, nếu hai người có nhiều con đường gián tiếp kết nối đến nhau, được điều chỉnh theo hệ số giảm dần, điểm số Katz của họ sẽ cao, cho thấy khả năng kết nối trực tiếp trong tương lai có thể cao.

- **Nhóm các phương pháp khác:**

a. **Bigram**

– **Mô tả**: Phương pháp Bigram thường được sử dụng trong phân tích ngữ nghĩa và xử lý ngôn ngữ tự nhiên. Trong ngữ cảnh dự đoán liên kết, Bigram có thể được áp dụng để dự đoán khả năng kết nối dựa trên sự xuất hiện liên tục của các nút trong một chuỗi các sự kiện hoặc tương tác.

– **Công thức**:

$$P(u,v) = \frac{Count(u,v)}{Count(u)}$$

Trong đó, $Count(u,v)$ là số lần $u$ xuất hiện ngay trước $v$, và $Count(u)$ là số lần $u$ xuất hiện.

– **Ví dụ**: Trong mạng giao tiếp email, nếu một người thường xuyên gửi email cho người khác theo một chuỗi, khả năng hai người này sẽ tiếp tục trao đổi email trong tương lai có thể cao.

b. **Clustering (Gom Nhóm)**

– **Mô tả**: Phương pháp gom nhóm (clustering) dự đoán liên kết dựa trên việc phân nhóm các nút trong mạng thành các cụm (clusters) và xác định khả năng kết nối giữa các nút thuộc các cụm khác nhau. Nút trong cùng một cụm có xu hướng liên kết với nhau hơn là với các nút ngoài cụm.

- **Công thức**: Một cách đơn giản để dự đoán liên kết dựa trên gom nhóm là:

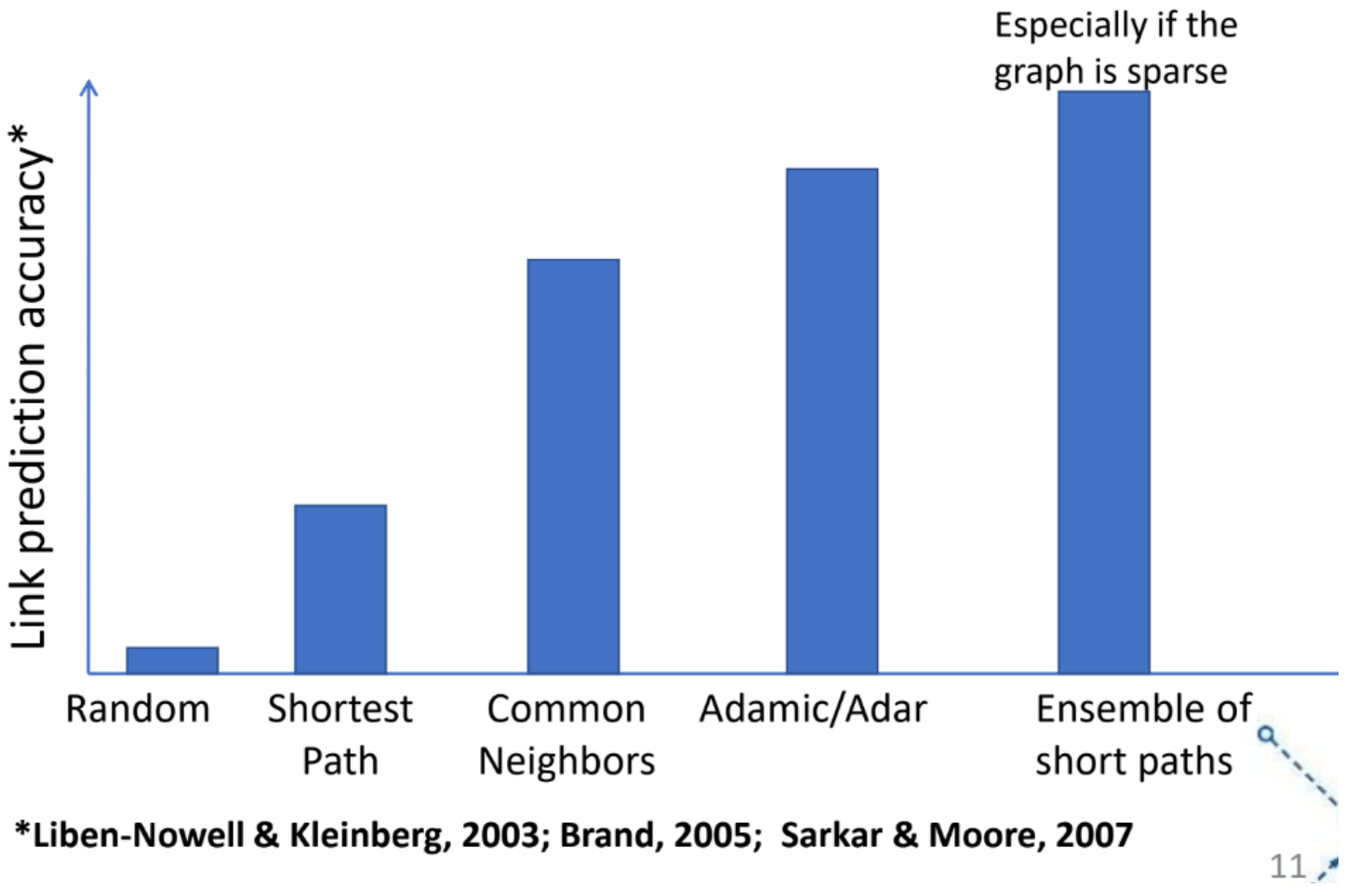
$$Score(u,v) = \text{similarity(cluster(u), cluster(v))}$$

  Trong đó, $cluster(u)$ và $cluster(v)$ là các cụm mà nút $u$ và $v$ thuộc về, và *similarity* đo lường sự tương đồng giữa các cụm.

- **Ví dụ**: Trong mạng xã hội, nếu hai người thuộc cùng một nhóm bạn bè hoặc nhóm sở thích, khả năng họ sẽ kết nối trực tiếp trong tương lai có thể cao hơn.

**So Sánh Hiệu Quả Của Các Phương Pháp Dự Đoán Liên Kết**

Để hiểu rõ hơn về hiệu quả của các phương pháp dự đoán liên kết trong mạng lưới, chúng ta hãy thực hiện phân tích so sánh mức độ chính xác của từng phương pháp. Bức hình dưới đây cung cấp cái nhìn sâu sắc về hiệu quả của các phương pháp khác nhau, bao gồm dự đoán ngẫu nhiên (Random), đường ngắn nhất (Shortest Path), số lượng láng giềng chung (Common Neighbors), phương pháp Adamic/Adar, và tập hợp các đường ngắn (Ensemble of short paths). Phân tích này giúp làm rõ các yếu tố ảnh hưởng đến độ chính xác của dự đoán liên kết và xác định các phương pháp có khả năng cung cấp dự đoán tốt nhất trong các tình huống khác nhau.




Hình 16: So Sánh Hiệu Quả Của Các Phương Pháp Dự Đoán Liên Kết

Hình 16 minh họa mức độ chính xác của một số phương pháp dự đoán liên kết (link prediction) trong mạng lưới, được trích từ các nghiên cứu của Liben-Nowell and Kleinberg (2003), Brand (2005), và Sarkar and Moore (2007). Trục dọc biểu thị độ chính xác của dự đoán liên kết, trong khi trục ngang liệt kê các phương pháp dự đoán khác nhau. Từ hình này chúng ta có thể thấy rằng:

- **Random:** Phương pháp dự đoán liên kết ngẫu nhiên có độ chính xác thấp nhất, hầu như không có tính hữu ích trong dự đoán.
- **Shortest Path:** Phương pháp sử dụng độ dài đường ngắn nhất giữa hai nút để dự đoán liên kết có độ chính xác cao hơn phương pháp ngẫu nhiên, nhưng vẫn khá thấp. Điều này có thể là do khoảng cách giữa các nút không phải là yếu tố quyết định duy nhất trong việc xác định liên kết

tiềm năng.

- **Common Neighbors:** Phương pháp này dựa trên số lượng láng giềng chung giữa hai nút. Đây là một phương pháp đơn giản nhưng khá hiệu quả, như thể hiện trong đồ thị, khi nó mang lại mức độ chính xác cao hơn đáng kể so với hai phương pháp trước.
- **Adamic/Adar:** Phương pháp này cải tiến từ phương pháp "Common Neighbors"bằng cách tính toán trọng số của các láng giềng chung, với trọng số cao hơn được gán cho những láng giềng ít phổ biến hơn. Kết quả là, độ chính xác của phương pháp này còn cao hơn phương pháp "Common Neighbors".
- **Ensemble of Short Paths:** Đây là phương pháp có độ chính xác cao nhất trong bức hình, đặc biệt là đối với các mạng lưới thưa (sparse graphs). Điều này cho thấy việc sử dụng một tập hợp (ensemble) các đường ngắn giữa các nút có thể cung cấp thông tin toàn diện hơn về mối quan hệ tiềm năng, dẫn đến dự đoán chính xác hơn.

Tóm lại, hình 16 cho thấy rằng các phương pháp dự đoán liên kết dựa vào đỉnh láng giềng cũng như dựa trên thông tin cấu trúc mạng như "Common Neighbors"và "Adamic/Adar"thường hoạt động tốt hơn so với các phương pháp đơn giản như "Random"hay "Shortest Path". Đặc biệt, phương pháp "Ensemble of Short Paths"thể hiện hiệu quả vượt trội, đặc biệt trong các mạng lưới thưa, nhấn mạnh tầm quan trọng của việc sử dụng thông tin từ nhiều đường ngắn để cải thiện độ chính xác của dự đoán liên kết.

# Tài liệu tham khảo


[1] DeepWalk: Online Learning of Social Representations (Trích: https://arxiv.org/pdf/1403.6652 )

[2] node2vec: Scalable Feature Learning for Networks (Trích: https://cs.stanford.edu/~jure/pubs/node2vec-kdd16.pdf )

[3] Mining graph patterns. (2016) Davide Mottin, Konstantina Lazaridou (Trích: https://hpi.de/fileadmin/user_upload/fachgebiete/mueller/courses/graphmining/GraphMining-04-FrequentSubgraph.pdf )

[4] A Survey of Frequent Subgraph Mining Algorithms. Chuntao Jiang, Frans Coenen and Michele Zito (Trích: https://cgi.csc.liv.ac.uk/~frans/PostScriptFiles/ker-jct-6-May-11.pdf )

[5] Frequent Subgraph Mining Algorithms – A Survey. (2015) T.Ramraja, R.Prabhakarb (Trích: https://core.ac.uk/download/pdf/82599177.pdf )

[6] Wikipedia contributors. (2024, August 12). Link prediction. Wikipedia. (Trích: https://en.wikipedia.org/wiki/Link_prediction )