Static Graph Challenge: Subgraph Isomorphism

Discrete Structures for Computer Science

Mai Hải Sơn - Nguyễn Đặng Anh Tài -
Trương Thành Tâm - Nguyễn Hoàng Sơn - Trần Minh Tâm

University of Technology - VNU-HCM
Faculty of Computer Science and Engineering

1st January 2024

Outline

Introduction

- Subgraph isomorphism is a computationally hard problem in graph theory and computer science.
- Subgraph queries enable matching semantic patterns and structures within complex graph datasets.
- However, solving subgraph isomorphism involves exponentially complex search spaces and graph comparison.
- The 2022 Static Graph Challenge provides a benchmark for subgraph isomorphism algorithms on massive real-world graphs in order to push the boundaries of what is computationally feasible today. In this report, we study subgraph isomorphism in the context of the 2022 Static Graph Challenge.

Community Detection and Applications

- **Introduction**
  Social network communities are tightly connected groups with shared interests or characteristics, forming dense clusters within a network. Examples include friends, colleagues, or individuals with similar hobbies.
- **Algorithms/Methods for Community Detection**
  - Graph Density Analysis: Identifies communities through dense subgraphs.
  - Modularity Detection: Measures network division strength for robust community structure.
  - Machine Learning: Uses k-means, spectral clustering for unsupervised community detection.
- **Applications**
  Community Detection for Targeted Insights: Unveil user clusters on social media for effective marketing, with broad applications in cybersecurity and social sciences.
- **Challenges and Future Trends**
  Confronting big data hurdles, privacy constraints, and diverse user communities. Future trends emphasize efficient algorithms, AI integration, and advancements in security and privacy tech.

Preliminaries

- Graph
  Graphs are basic structures with vertices and edges, varying in terms of edge direction, multiple edges between vertices, and the presence of loops.
- Adjacency Matrices
  Suppose that $G = (V, E)$ is a simple graph where $|V| = n$. In other words, if its adjacency matrix is $A = [a_{ij}]$, then

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

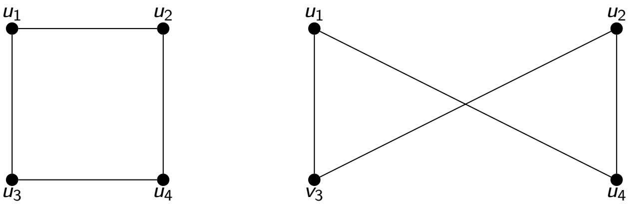
- Incidence Matrices
  Let $G = (V, E)$ be an undirected graph. The incidence matrix with respect to this ordering of V and E is the $n \times m$ matrix $M = [m_{ij}]$, where

$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

Preliminaries

- Isomorphism of Graphs
  Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a bijective function $f : V_1 \to V_2$ such that adjacency in G_1 corresponds to adjacency in G_2 under f. The function f is called an isomorphism. Nonisomorphic graphs lack such a function.

Example: $G = (V, E)$ and $H = (W, F)$ are isomorphic.



$f : U \to V$ with
$f(u_1) = v_1, \quad f(u_2) = v_4, \quad f(u_3) = v_3, \quad f(u_4) = v_2$

Triangle Counting and k-Truss Algorithms

Algorithm 1: Triangle Counting

Data: Adjacency matrix A and incidence matrix E

$C = AE$;

$n_t = nnz(C)/3$;

Multiplication is overloaded such that

$C(i,j) = f(i,x,y) \iff A(i,x) = A(i,y) = 1$ & $E(x,j) = E(y,j) = 1$;

Result: Number of triangles in graph G initialization;

Algorithm 2: k-Truss

Data: Unoriented incidence matrix E and integer k

$d = \text{sum}(E)$;

$A = E^T E - \text{diag}(d)$;

$R = EA$;

$s = (R == 2)1$;

$x = \text{find}(s < k - 2)$;

while *x is not empty* **do**

    $E_x = E(x, :)$;

    $E = E(x_c, :)$;

    $d_x = \text{sum}(E_x)$;

    $R = R(x_c, :)$;

    $R = R - E[E_x^T E_x - \text{diag}(d_x)]$;

    $s = (R == 2)1$;

    $x = \text{find}(s < k - 2)$;

end

Result: Incidence matrix of k-truss subgraph E_k initialization;

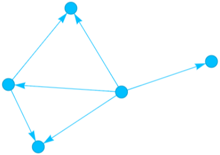
Ullman and VF2 algorithm

- Ullman algorithm
  Ullman algorithm is the earliest and highly-cited approaches to the subgraph isomorphism problem is the algorithm proposed by Ullman.
- VF2 algorithm
  VF2 is an improvement on the Ullman algorithm. The VF2 algorithm starts with an empty mapping and iteratively extends it while considering consistency and cutting functions until a complete mapping is achieved.

Algorithm 3: VF2

procedure VF2(Mapping m, ProblemType PT)
if m covers V_1 **then**
    Output(m)
else
    Compute the set P_m of candidate pairs for extending m
    for all $p \in P_m$ **do**
        if ConsPT$(p, m) \wedge \neg$CutPT(p, m) **then**
            call VF2(extend(m, p, PT))

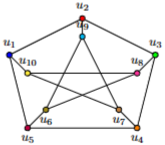
Triangle Counting



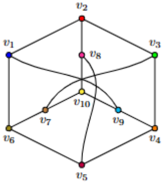
Result

```
Counting triangles
Number of triangles in   : 2.0
Execution completed.

```

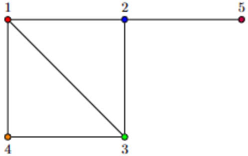


G



H

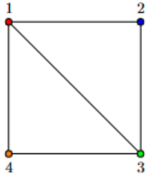
The two isomorphic graphs above both have the number of triangles equal to 0.



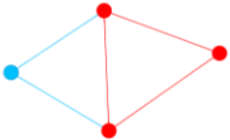
$\xrightarrow{\text{Result}}$

$$E = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

It corresponds to the graph:



VF2 Algorithm

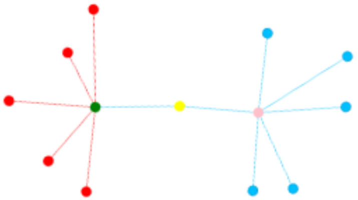


subgraph_isomorphisms = list(nx.algorithms.isomorphism.GraphMatcher(my_graph, pattern_graph).subgraph_isomorphisms_iter())

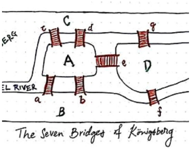
```
There are subgraph isomorphism
{1: 1, 2: 2, 3: 3}
{1: 1, 3: 2, 2: 3}
{1: 1, 3: 2, 4: 3}
...
{3: 1, 4: 2, 1: 3}
{4: 1, 1: 2, 3: 3}
{4: 1, 3: 2, 1: 3}
Number of subgraph isomorphisms: 12
<IPython.core.display.HTML object>
```

```
G = nx.DiGraph() #Load directed graph
subgraph_isomorphisms=
list(nx.algorithms.isomorphism.DiGraphMatcher(my_graph,
pattern_graph).subgraph_isomorphisms_iter())
```

We can find the isomorphic subgraph of a graph H in the larger graph G.
It can be applied in determining the community in a social network.
For example, we have a community of students studying discrete structures and calculus 2 in the year 2023.



The Königsberg Bridges Problem queries if there is a single continuous walk crossing each of the seven bridges exactly once, and if a closed walk is possible.



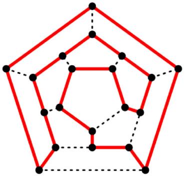
Euler solved the Königsberg bridges problem using a letter sequence (A, B, C, D) and introduced the "handshaking lemma" for graphs.
Euler's conclusions for the general situation were:

- If more than two areas have an odd number of bridges, a journey is impossible.
- If exactly two areas have an odd number of bridges, the journey is possible starting from either.
- If no areas have an odd number of bridges, the journey is feasible from any starting point.

In 1878, James Sylvester coined the term "graph" and introduced cycle concepts, contributing to 19th-century graph theory.

The Königsberg Bridges

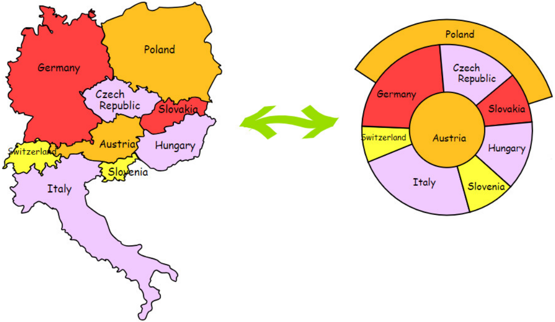
- Tree counting
  Arthur Cayley (1821–1895) contributed to graph theory by studying tree enumeration, defining spanning trees, and inventing a method for counting unrooted trees, applied to chemical molecules.
- Icosian Game
  Sir William Rowan Hamilton's icosian game posed a graph theory puzzle, finding a Hamiltonian cycle on a dodecahedron. It's an early formulation of the Traveling Salesman Problem.



- Four-Color Theorem
  Francis Guthrie (1831–1899) posed the Four-Color Problem in 1852. Alfred Bray Kempe initially claimed proof in 1879, later corrected by Percy John Heawood in 1890. The theorem remained unsolved until the late 20th century, aided by computer advancements.

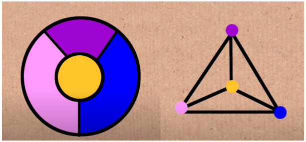
The theorem

The Four Color Theorem, proposed in 1852 by Francis Guthrie, was proven in 1976 by Kenneth Appel and Wolfgang Haken using computer-assisted methods, marking a culmination of centuries-long mathematical exploration and conjectures.
The Four Color Theorem asserts that any map on a plane can be colored using only four colors, ensuring that adjacent regions (excluding single points) have distinct colors.



The first proof by Kempe

In 1879, Alfred Kempe presented the first widely accepted proof for the Four Color Theorem, analyzing country connections by representing maps as planar graphs with vertices and edges.



Kempe discovered that all maps, regardless of complexity, contain at least one region with at most five neighbors, termed "unavoidable sets."
Kempe's analysis found a vertex with at most five neighbors in every simple planar graph.

- Six special configurations were identified based on this finding.
- Removing a vertex and its edges left a four-colorable graph with n vertices.
- If adjacent vertices had three or fewer colors, the removed vertex could be colored, completing the four-coloring.

Kempe's recoloring method, proposed for vertices with all four colors, was later found to have errors by Percy Heawood in 1890.

Proof by Computer

Heinrich Heesch used discharging rules to find an unavoidable set, assigning and redistributing charges in a graph. This method ensures the existence of an unavoidable set in map graphs.
In 1976, Kenneth Appel and Wolfgang Haken at UIUC proved the four-color theorem using dynamic programming. They identified a set of 1,936 reducible configurations, proving their four-colorability. This dynamic programming approach extended the proof to other configurations within the set.