Subgraph Isomorphism

Artem Maksov, Yong Li, Reese Butler
03/04/2015
Similar Graphs

- The two graphs below look different but are structurally the ‘same’.
What is Graph Isomorphism?

An isomorphism of graphs $G$ and $H$ is a bijection between the vertex sets of $G$ and $H$

$$f: V(G) \to V(H)$$

such that any two vertices $u$ and $v$ of $G$ are adjacent in $G$ if and only if $f(u)$ and $f(v)$ are adjacent in $H$.

This kind of bijection is generally called "edge-preserving bijection", in accordance with the general notion of isomorphism being a structure-preserving bijection.

If an isomorphism exists between two graphs, then the graphs are called isomorphic and we write $G \simeq H$.
### One Isomorphism Example

#### Graph G

- Node a
- Node b
- Node c
- Node d
- Node g
- Node h
- Node i
- Node j

#### Graph H

- Node 1
- Node 2
- Node 5
- Node 6
- Node 8
- Node 7
- Node 3

#### An isomorphism between G and H

- \( f(a) = 1 \)
- \( f(b) = 6 \)
- \( f(c) = 8 \)
- \( f(d) = 3 \)
- \( f(g) = 5 \)
- \( f(h) = 2 \)
- \( f(i) = 4 \)
- \( f(j) = 7 \)
Basic Requirement

- **Requirement for Graph Isomorphism**
  - The same number of vertices
  - The same number of edges
  - Structural similarity
    - Four-regular graph
    - Bipartite graph
Sometimes it is easy to check whether two graphs are not isomorphic.

- Different number of vertices
- Different number of edges
- Structural difference
It is much harder to prove that two graphs are isomorphic. The computation in time is exponential wrt. the number of vertices.

- Problem is in NP, but
  - No NP-completeness proof is known
  - No polynomial time algorithm is known
- If GI is NP-complete, then “strange things happen”
  - “Polynomial time hierarchy collapses to a finite level”

If $P \neq NP$
History

- The problem of determining isomorphisms between a set of graphs first became known during the 1960’s as a method of comparing two chemical structures.
- A computational enigma - In 1979 listed by Garey & Johnson as one in 12 problems belonging to NP, but not known to be either NP-complete or solvable in polynomial time.
- Best (theoretical) algorithm was created by Eugene Luks in 1983.
Applications

• Pattern matching in various applications:
  • Cheminformatics
  • Bioinformatics
  • Databases
  • Design of Circuits
  • Image Processing and Computer Vision
  • Crystallography
  • Social Networks Analysis

• Planar isomorphism is a natural match for data structures embedded in a plane:
  • City planning
  • Analysis of material surface
Cheminformatics

- Analysis of complex molecular structures
- Can deal with molecules consisting of thousands of atoms
- Even minor differences in structure can lead to significant differences in properties
- Important for material and drug design
- Maximum common subgraph problem is the most relevant
- Most molecules can be represented as planar graphs to reduce complexity

(Raymond, Willett; 2002)
Bioinformatics

- Same issues as cheminformatics but on a larger scale
- Applied mostly to proteins
- Properties of proteins depend not only on general structure, but on 3D folding
- Can be studied with general isomorphism algorithms or specialized ones

(S. VISHVESHWARA, K. V. BRINDA, N. KANNAN, 2002)
Crystallography

- X-ray diffraction technique discovered that crystalline solids consist of atoms arranged in an orderly repetitive arrays.

- Focus in modern science shifted from characterization of a crystal to prediction of its physical properties from first principles calculations before growing the crystal.

- Both 3D and 2D structures can be viewed as crystals.
Periodic 3D structures characterized by a repeatable building block

Simple Cubic

Body-Centred Cubic

Face-Centred Cubic

Can be much more complex

Diamond Cubic  Zincblende  Triclinic  Hexagonal

What is the mathematical nature of crystal structures?

“Topologically they are infinite-fold abelian covering graphs over finite graphs. Crystals are their periodic realizations in space.”

- attributed to Henri Poincaré

“In abstract algebra, a **free abelian group** or **free Z-module** is an abelian group with a basis. That is, it is a set together with an associative, commutative, and invertible binary operation, and its basis is a subset of its elements such that every element of the group can be written in one and only one way as a linear combination of basis elements with integer coefficients, finitely many of which are nonzero.”

- Wikipedia, “Free Abelian Group”

(Sunada, 2012)
What is the mathematical nature of crystal structures?

- Fundamental finite graphs can be obtained by factoring out translational elements from infinite crystal net

(Sunada, 2012)
What is the mathematical nature of crystal structures?

Given a finite graph $X_0 = (V_0, E_0)$ and $X = (V, E)$:

Morphism $\omega: X \rightarrow X_0$ is a correspondence of vertices and edges preserving the adjacency relations between them.

Bijective $\omega$ is an Isomorphism

$\omega: X \rightarrow X$ is an Automorphism
Real space imaging

- Recently allows to image atoms directly
- Scanning Transmission Electron Microscopy (STEM) and Scanning Tunneling Microscopy and non-contact Atomic Force Microscopy
- Planar structures by the nature of technology, but periodicity is broken due to defects and noise
- Edges can not be imaged
Graph reconstruction from real space images

1. Extract atomic positions from the image

Knowing the types of unit cells present in the crystal lattice would help to determine the properties of the material, quantify the defects in the crystal lattice, and, potentially, advance our knowledge base for the Materials Genome Initiative.
2. Calculate Radial Distribution Function: a histogram of distances from each atom to their neighbors within certain distance

3. From the histogram determine the edge types
4. Connect vertices with weighted edges

5. With 2 types of edges, 6 possible unit cell possible:
   a. 4 V, 4E1
   b. 4V, 4E2
   c. 4V, 3E1, 1E2
   d. 4V, 3E2, 1E1
   e. 4V, 2E1, 2E2 adjacent
   f. 4V, 2E1, 2E2 alternating

Subgraph isomorphism would help to find all combinations
Algorithms

- **VF2** - begins with first vertex, selects sequential vertices, checks if subgraph matches, backtracks if not
- **QuickSI** - designed for handling small graphs, often outperforms the more recent algorithms such as GraphQL, GADDI, and SPath which are intended for use with large graphs
  - attempts to select vertices which have infrequent vertex labels and infrequent, adjacent edge labels as early as possible
- **GraphQL** - uses neighborhood signature based pruning
- **GADDI** - relies on the Neighboring Discriminating Substructure (NDS) distance
- **SPath** - relies on generating neighborhood signatures in the graph to minimize candidate sets
- All the existing algorithms have problems with their join order selections for at least some datasets
A Brute-force Solution

• Given a Large graph $G_1(V_1, E_2)$, Small graph $G_2(V_2, E_2)$
• Select a subset of vertices from $G_1$, denote as $G_1'(V_1', E_1')$, where $|V_1'| = |V_2|$ (Combination = $C(|V_1|, |V_2|)$)
• Match the vertices in $G_1'$ to the vertices in $G_2$ (Permutation = $|V_2|!$)
• Check if the permuted vertices of $G_1'$ is isomorphic to $G_2$
• Complexity:
  $C(|V_1|, |V_2|) \times |V_2|! = |V_1|! / (|V_1| - |V_2|)!$
VF2

- **State Space Representation (SSR):** Each state \( s \) of the matching process can be associated to a partial mapping solution \( M(s) \), where \( M(s) \) contains isomorphic subgraphs from \( G_1 \) and \( G_2 \) and each vertex of \( G_1 \) in \( M(s) \) is mapped to a vertex of \( G_2 \) in \( M(s) \).

- **Complexity:**
  - **Spatial:** \( O(V) \), where \( V \) is the (maximum) number of vertices of the two graphs
  - **Time:** Time complexity is \( O(V^2) \) in the best case and \( O(V! \cdot V) \) in the worst case.
1. Match empty $G_1$ and $G_2$ - always works
2. Match $1G_1$ with $1G_2$ - always works
3. Match $2G_1$ with $2G_2$ - works because
   $\{1, 2\}$ of $G_1$ is isomorphic to $\{1, 2\}$ of $G_2$
4. Match $3G_1$ with $3G_2$ - fails because
   $\{2, 3\}$ of $G_1$ is not in $G_2$, go back to step 2
5. Match $2G_1$ with $3G_2$ - works because
   $\{1, 2\}$ of $G_1$ is isomorphic to $\{1, 3\}$ of $G_2$
6. Match $3G_1$ with $2G_2$ - fails because
   $\{2, 3\}$ of $G_1$ is not in $G_1$, go back to step 1
7. Match $1G_1$ with $2G_2$ - works
8. Match $2G_1$ with $1G_2$ - works
9. Match $3G_1$ with $3G_2$ - works.
PROCEDURE Match(s)
    INPUT: an intermediate state $s$; the initial state $s_0$ has $M(s_0) = \emptyset$
    OUTPUT: the mappings between the two graphs

    IF $M(s)$ covers all the nodes of $G_2$ THEN
        OUTPUT $M(s)$
    ELSE
        Compute the set $P(s)$ of the pairs candidate for inclusion in $M(s)$
        FOREACH $p$ in $P(s)$
            IF the feasibility rules succeed for the inclusion of $p$ in $M(s)$ THEN
                Compute the state $s'$ obtained by adding $p$ to $M(s)$
                CALL Match($s'$)
            END IF
        END FOREACH
        Restore data structures
    END IF
END PROCEDURE Match
Implementations

Elapsed time for the algorithms run on test dataset:

(a) subgraph queries.  
(b) clique queries (Y-axis in log scale).  
(c) path queries.
Comparison of execution time with brute-force way and VF2 (using library from [http://mivia.unisa.it/datasets/graph-database/vflib/](http://mivia.unisa.it/datasets/graph-database/vflib/)) in milliseconds

<table>
<thead>
<tr>
<th>Large graph size</th>
<th>Small graph size</th>
<th>Brute-force</th>
<th>VF2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>767</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>50,502</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>11</td>
<td>2,700,000+</td>
<td>1</td>
</tr>
<tr>
<td>161</td>
<td>156</td>
<td>-(more than 2h)</td>
<td>153</td>
</tr>
<tr>
<td>1161</td>
<td>1156</td>
<td>-</td>
<td>53,781</td>
</tr>
</tbody>
</table>
Open Problems/Future Directions

- Determine if graph isomorphism problem is NP-complete or solvable in polynomial time.
- Reduce base run-time complexity of algorithms
- Improve vertex selection pruning
1. L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs.
1. Determine if the following pairs of graphs are isomorphic.

a. 

b. 

2. For each of the following, find a subgraph in A which is isomorphic with B.

Part 1: A.

Part 2: A.

3. How many unique (ignoring differing label combinations) subgraph isomorphisms are in part 1? Part 2?