Speeding Up GED Verification for Graph Similarity Search

Lijun Chang

Joint work with Xing Feng (Google), Xuemin Lin (UNSW), Lu Qin (UTS), Wenjie Zhang (UNSW), Dian Ouyang (USYD)

April 22, 2020
Graph Similarity Search

Given a database $\mathcal{D} = \{g_1, g_2, g_3, \ldots\}$ consisting of a set of vertex and/or edge labeled graphs, graph similarity search aims to find all graphs in $\mathcal{D}$ that are similar to a user-given query graph $q$.

- Here, inexact/similarity search is used
- Because exact graph search may find no or very few results due to erroneous data entry, data noise or nature of the application
Graph edit distance (GED) is a widely used distance/similarity measure in graph similarity search studies.\footnote{Xiang Zhao et al. “A Partition-Based Approach to Structure Similarity Search”. In: PVLDB 7.3 (2013).} \footnote{Yongjiang Liang and Peixiang Zhao. “Similarity Search in Graph Databases: A Multi-Layered Indexing Approach”. In: Proc. of ICDE’17. 2017.} \footnote{Xiang Zhao et al. “Efficient structure similarity searches: a partition-based approach”. In: VLDB J. 27.1 (2018).} \footnote{Jongik Kim, Dong-Hoon Choi, and Chen Li. “Inves: Incremental Partitioning-Based Verification for Graph Similarity Search”. In: Proc. of EDBT’19. 2019.}

- GED is a metric
  - Applicable to all types of graphs
  - Captures the structural difference between graphs
- $\text{ged}(q, g)$: minimum number of edit operations needed to transform $q$ into $g$
  - Vertex/Edge relabeling
  - Edge insertion/deletion
  - (Isolated) vertex insertion/deletion
Graph Edit Distance

\[ \text{ged}(q, g) = 5 \]

- The following is a sequence of 5 edit operations that transform \( q \) into \( g \)

(1) Relabel \( v_1 \) to 'B'

(2) Relabel \( (v_2, v_3) \) to 'b'

(3) Add \( v_5 \) with label 'C'

(4) Add \( (v_1, v_5) \) with label 'b'

(5) Add \( (v_4, v_5) \) with label 'a'
Filtering-and-Verification

Formally, the graph similarity search problem is to compute
\( \{ g \in \mathcal{D} \mid \text{ged}(q, g) \leq \tau \} \) for user-specified \( q \) and \( \tau \)
- A naive approach is checking, for every \( g \in \mathcal{D} \), whether \( \text{ged}(q, g) \leq \tau \)
- This is expensive as deciding whether \( \text{ged}(q, g) \leq \tau \) is NP-complete

Filtering-and-verification paradigm.
1. Candidate generation: \( \text{cand} \subseteq \mathcal{D} \)
   - \( \text{ged}(q, g) > \tau \) for every \( g \in \mathcal{D} \setminus \text{cand} \)
   - Filter out unpromising data graphs (possibly by probing an offline-constructed index)
   - Based on pigeonhole principle: if there are \( \tau + 1 \) disjoint substructures (e.g., path, tree, subgraph) of \( q \) not appearing in \( g \), then \( \text{ged}(q, g) > \tau \)
2. Candidate verification
   - Verify whether \( \text{ged}(q, g) \leq \tau \), for every \( g \in \text{cand} \)
Our Contribution: Speeding Up GED Verification

- The existing studies focus on generating a small candidate set (by designing different index structures), while using an outdated algorithm $A^*_{GED}$ for GED verification.

- We propose an efficient algorithm $AStar^+-LSa$ to speed up GED verification, which is orthogonal to the existing indexing/filtering techniques.

- Our experimental results show that
  - The existing indexing/filtering techniques either have very limited filtering power or take a very long filtering time (e.g., may even longer than directly verifying all data graphs by $AStar^+-LSa$).
  - Thus, the existing indexing/filtering techniques become obsolete given our efficient GED verification algorithm $AStar^+-LSa$. 

GED Computation Via Vertex Mapping

- \( \text{ged}(q, g) \) can be computed by enumerating vertex mappings from \( q \) to \( g \).
  - Vertex insertion can be encoded by mapping a dummy vertex to \( V(g) \).
  - Vertex deletion can be encoded by mapping \( V(q) \) to a dummy vertex.

A search tree \( T \) compactly represents all vertex mappings from \( V(q) \) to \( V(g) \): \( f_i \) is a partial mapping, and beside \( f \) at level \( j \) is a pair \((u, \text{lb}_f)\) where \( u \in V(g) \) is the vertex to which \( v_j \) maps and \( \text{lb}_f \) is a lower bound of \( f \).
Our GED Verification Algorithm \( \text{AStar}^+ - \text{LSa} \)

- \( \text{AStar}^+ - \text{LSa} \) conducts a best-first search of the search tree \( \mathcal{T} \), based on lower bounds \( \text{lb}_f \) of partial mappings \( f \)
  - \( \text{AStar}^+ - \text{LSa} \) uses a fixed matching order of \( V(q) \)

- The efficiency of \( \text{AStar}^+ - \text{LSa} \) is achieved by three ingredients
  1. Don’t need to add dummy vertices to \( q \) or \( g \)
  2. Tighter lower bound estimation
  3. Efficient lower bound computation
Ingredient 1: Don’t Add Dummy Vertices

We prove that if $|V(q)| \leq |V(g)|$, then there is no vertex deletion in the optimal sequence of edit operations that transform $q$ into $g$

W.l.o.g., we assume that $|V(q)| = |V(g)|$
- If $|V(q)| < |V(g)|$, then we can add $|V(g)| - |V(q)|$ dummy vertices to $q$
- Thus, we don’t need to consider vertex insertion/deletion
- In implementation, we don’t add dummy vertices to $q$ even if $|V(q)| < |V(g)|$

Advantages of not considering vertex insertion/deletion
- Reduces the number of full mappings from $\approx (|V(g)| + 1)|V(q)| + |V(g)|$ to $|V(g)||V(q)|$
- Simplifies algorithm implementation
Consider the partial mapping \( f = \{v_1 \mapsto u_1, v_2 \mapsto u_2\} \).

The existing algorithms use label set-based lower bound \( \text{lb}_{LS}^f \):

- \( mc_f \): the number of edit operations required to transform \( q_f \) into \( g_f \) by obeying \( f \)
- The vertex (resp. edge) label difference between the unmapped parts \( q_{\bar{f}} \) and \( g_{\bar{f}} \)
- \( \text{lb}_{LS}^f = mc_f + \Upsilon (L_V(q_{\bar{f}}), L_V(g_{\bar{f}})) + \Upsilon (L_E(q_{\bar{f}}), L_E(g_{\bar{f}})) = 1 + \Upsilon (\{A, B, C\}, \{A, A, E\}) + \Upsilon (\{a, a, b\}, \{a, a, a\}) = 4 \)
We propose anchor-aware label set-based lower bound $\text{lb}^{\text{LSa}}_f$ by separating the cross edges from the unmapped parts: $\text{lb}^{\text{LSa}}_f = \text{mc}_f +$

- $\Upsilon(L_{EC}(v_1), L_{EC}(u_1)) +: \Upsilon(\{b\}, \{\}) = 1$
- $\Upsilon(L_{EC}(v_2), L_{EC}(u_2)) +: \Upsilon(\{a\}, \{a\}) = 0$
- $\Upsilon(L_{E_I}(q_f), L_{E_I}(g_f)) +: \Upsilon(\{a\}, \{a, a\}) = 1$
- $\Upsilon(L_{V}(q_f), L_{V}(g_f)): \Upsilon(\{A, B, C\}, \{A, A, E\}) = 2$
- $\text{lb}^{\text{LSa}}_f = 5 > \text{lb}^{\text{LS}}_f = 4$

We prove that $\text{lb}^{\text{LSa}}_f \geq \text{lb}^{\text{LS}}_f$ holds for any mapping $f$
Ingredient 3: Efficient Lower Bound Computation

- In the best-first search, for a partial mapping \( f \), we need to compute the lower bound for all children \( h \) (i.e., one-vertex extension) of \( f \).

- The existing works compute the lower bound for each child \( h \) independently:
  - Total time complexity of \( \mathcal{O}(|V(g)| \times (|E(q)| + |E(g)|)) \).

- We propose an algorithm with total time complexity of \( \mathcal{O}(|E(q)| + |E(g)|) \), by online constructing a data structure and conducting computation incrementally.
Experimental Setting

- **Datasets**
  - AIDS: an antivirus screen chemical compound dataset published by the Developmental Therapeutics Program at NCI/NIH \(^5\)
  - PubChem: a chemical compound dataset \(^6\)

<table>
<thead>
<tr>
<th>Database</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIDS</td>
<td>42,689</td>
<td>25.6</td>
<td>27.5</td>
<td>222</td>
<td>247</td>
<td>66</td>
<td>3</td>
</tr>
<tr>
<td>PubChem</td>
<td>23,903</td>
<td>48.3</td>
<td>50.8</td>
<td>88</td>
<td>92</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

- All algorithms are run in main memory, and run as single-thread algorithms

\(^5\)https://cactus.nci.nih.gov/download/nci/AID2DA99.sdz
Index-free Graph Similarity Search

- **Algorithms**
  - AStar⁺-LSa: our algorithm
  - CSI_GED\(^7\): depth-first search + edge mapping
  - Inves\(^8\): online graph partitioning-based filtering
- **To verify** \(\text{ged}(q, g) \leq \tau\), all the three algorithms first run LabelF for filtering
  - That is, if the label-set based lower bound is larger than \(\tau\), then \(g\) is pruned

---


Index-based Filtering for Graph Similarity Search

- **Filtering time ratio of Pars**: \( \frac{\text{filtering time of Pars}}{\text{total running time of AStar}^+ - \text{LSa}} \)

- **Filtered candidate ratio of Pars**: \( \frac{\text{number of candidates filtered by Pars}}{\text{total number of candidates generated by LabelF}} \)

Filtering effectiveness of Pars

\[ \tau = \text{Filtering time ratio} \]

\[ \tau = \text{Filtered cand ratio} \]

\[ \tau = \text{Filtering Power (\%)} \]

---

**Our Algorithms for Graph Similarity Search**

Processing time of our algorithms for 100 random queries

- AStar$^+$-LSa and DFS$^+$-LSa perform similarly
  - For graph similarity search, most of the pairs $(q, g)$ are dissimilar pairs
  - We show in the paper that for dissimilar pairs, best-first search and depth-first search have the same search space and thus similar running time
GED Computation

Processing time for GED computation (ged = 9)
Conclusion

► We proposed an efficient algorithm AStar\textsuperscript{+}-LSa to speed up GED verification, which is achieved by three ingredients
  – Don’t need to add dummy vertices to $q$ or $g$
  – Tighter lower bound estimation
  – Efficient lower bound computation

► The existing indexing/filtering techniques become obsolete given our efficient GED verification algorithm AStar\textsuperscript{+}-LSa

► The source code of our algorithms will be available at https://github.com/LijunChang/Graph>Edit_Distance.