GString : A Novel Approach for Efficient Search in Graph Databases (ICDE 2007)

Presented to CS 8790
By Michael Scott
Problem definition

- **Graph search** :

  “Given a graph database $D = \{g_1, g_2, \ldots, g_n\}$ and a graph query $q$, find all $g$ in $D$ such that $q$ is a subgraph of $g_i$.”

- **Subgraph isomorphism**
  - known NP-Complete
  - Must be done for each graph in database
The problem gets worse

- support approximate matching as well
  - “in real life applications such as searching for chemical compounds, subgraph isomorphism checking must tolerate some disparities.”

- “even small graphs may create a combinatorial explosion … translate[s] to a large index space”
Supporting graph queries

- “a popular approach is to represent both graphs and queries … by sequences”

- “state-of-the-art methods […] each node in the graph will appear in the sequence”
Introducing GString

- “introduce a novel sequencing method to capture the semantics of the underlying graph data”

- “we find meaningful components in graph structures and use them as the basic units in sequencing”

- “reduces size of resulting sequences”

- Preserves ‘meaning’
The authors propose a method for supporting “efficient search in graph databases”

“efficient search [of chemical compounds] in [a special subset of] graph databases”

Convert graphs (and queries) into string form

- Use existing methods for string indexing
Related Work

- **GraphGrep (ICPR 2002)**
  - ‘path based’
  - supports REX graph query lang

- **GIndex (SIGMOD 2004)**
  - Unique graph ‘fragments’
  - More expensive to compute
    - But yields smaller indexes

- **C-Tree (ICDE 2006)**
  - Closure-Tree, use graph closures
GraphGrep

- Enumerate paths of length $n$
  - Record # occurrences
  - Compare with a query

- Exponential number of potential paths
  - Must limit the length of paths

- “paths and fragments only carry local structure information of a graph.”
Example path decomposition (len 3)

![Diagram of a chemical structure labeled 2',3'-dideoxy-3'-fluorocytidine]
Gindex

- frequent graph patterns
  - “Discriminative fragments”

- Takes longer to precompute

- Reduction of index space

- Improves filtering rate
  - Retains more information than path-based deconstruction
C-Tree

- **Closure-Tree** – a tree-based indexing mechanism

- A node in the tree is the closure of the sub-graphs it represents

- Reduces index space

- This aggregation enhances pruning
Graph Closures

Figure 1. A Sample graph database

Figure 2. Sample graph closures
Limitations of other approaches

- **GraphGrep**
  - “combinatorial explosion”

- “not every path or fragment is meaningful” on its own, at least as far as an application may be concerned.

- “when we break down a graph we risk losing information about the ‘global structure’”
  - May “lead to huge numbers of false positives”

- no substructure similarity search
A weakness of path-based approaches

Figure 1: A Sample Database

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>c-c</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c-c-c</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>c-c-c-c</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: A Sample Query

Figures from gIndex paper

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>c</td>
<td>10</td>
</tr>
<tr>
<td>c-c</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>c-c-c</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>c-c-c-c</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
Limitations of other approaches

- Gindex
  - Although the discriminative fragments are an improvement
  - The authors note that “the similarity of two compounds is poorly captured by the enumerations of paths and fragments.”
  - “in order to reveal the similarity a lot of ‘join’ operations must be conducted”
  - “another dimension of combinatorial explosion”
  - no substructure similarity search
Graph String

- Convert graphs and queries to strings
  - Should represent the semantics of the graph

- Use string algorithms
  - Indexing
  - Comparison - similarity

- Specifically
  - Suffix tree
Basic Structures

- Line – vertices connected end-to-end
- Cycle – vertices form a closed loop.
- Star – core connected to several other vertices.

Figure 2. Basic Structures
Structure annotations

- node is by default a carbon,
  - non-carbon nodes annotated

- Edges are by default single bonds,
  - may be annotated as double bonds, triple..

- Branching annotations

- Hydrogen atoms inferred where possible
Figure 3. Annotation Examples
Table 1. Grammar for basic structures
Gstring examples

(a) $g_1$: 2',3'-dideoxy-3'-fluorocytidine

(b) $g_2$: Cytidine, 2',3'-dideoxy-2',5-difluoro

$G_1$:
- Line 2\{node 1 O\} (c 2 1)
- Cycle 5\{node 2 O\} (branch 5 F) (c 3 1)
- Cycle 6\{node 1 N\} (node 3 N) (edge 3 4 d) (edge 5 6 d)
  (branch 2 O) (branch 4 N)

$G_2$:
- Line 2\{node 1 O\} (c 2 1)
- Cycle 5\{node 2 O\} (branch 4 F) (c 3 1)
- Cycle 6\{node 1 N\} (node 3 N) (edge 3 4 d) (edge 5 6 d)
  (branch 2 O) (branch 4 N) (branch 5 F)
GString queries

- Goal: index on the features of a graph

- Extract certain features from gstring and index on them

- These features form a ‘summary string’
  - Allowing certain disparities

- In this domain
  - Line, cycle, and star are the important elements
Query Example

Figure 7. Query example

Example 3 (GString for the query graph in Fig. 7)
Cycle 5 \( \langle node \ 5 \ O \rangle \langle branch \ 2 \ F \rangle \langle c \ 1 \ I \rangle \)
Cycle 6 \( \langle node \ 1 \ N \rangle \langle node \ 3 \ N \rangle \langle edge \ 3 \ 4 \ d \rangle \langle edge \ 5 \ 6 \ d \rangle \langle branch \ 2 \ O \rangle \langle branch \ 4 \ N \rangle \langle branch \ 5 \ F \rangle \)
Summary strings

- Ideally should capture the meaning of the graph
  - While also permitting approximate matching

- Type and size of basic structures

- Information about the edits (with wiggle room)
Summary Strings

- A summary string will contain the type, size of each basic structure

- We record the number of node, branch, and edge annotations

- The idea is to index on the type/size of structures
  - Use the annotation counts later for additional filtering
Summary String (cont.)

- Type Size \[ n_n \ n_b \ n_e \]
- Generate the summary strings for each structure
- Concatenate results
Example Summary String

Summary string for this query:
Cycle 5 [1 1 0] Cycle 6 [2 3 2]
Indexing summary strings

- For each “extracted” path
  - Index reverse path as well

- Suffix-tree based indexing struct
  - 2 parts:
    - Suffix tree – indexes the type, size information
    - Edit table – stores the edit information (used for filtering later)

- Match conditions:
  - Type and size match (cycle 6 != line 6), (cycle 5 != cycle 6)
  - Then match on number of edits
Using suffix tree

- Suffix tree supports many useful string operations

- Check if a string $P$, where $|P| = m$ is a substring
  - $O(m)$

- Find all $z$ occurrences of patterns $P_1, \ldots, P_q$ where $|P_1| + \ldots + |P_q| = m$
  - $O(m + z)$

- Importantly: can support search efficiently with mismatches!
Graph simplification

(c) $g_3$: 3-ethyl-2-propyl-1,1’-bi(cyclohexane)-1,3’-diene

(a) Compound of Figure 4(c)
Experiment

- NCI/NIH AIDS Antiviral screen dataset
  - 43,000 chemical compounds

- Avg – 25 vertices, 27 edges

- Max – 222 vertices, 251 edges

- --not very ‘big’
Experiment Methodology

1 - Index size, construction time
   - Set db size to 5k, 10k, 20k, 30k, 40k
   - Measure idx size, construction time

2 - Subgraph query experiment
   - Fix db size to 20k
   - Query size = [8,24]
     - Randomly extract connected structures from graphs in db
Experiment Methodology (cont.)

- 3 - Similarity queries – just for GString
  - Fix db size to 5k

- Vary query sizes [8,24]
  - For each size, generate 500 queries of that length

- Query creation
  - Randomly choose a graph
  - Extract some connected structures
  - Randomly add branches, relabel nodes & branches
(a) Index size
(b) Construction time
(c) Candidate set size
(d) Accuracy
(e) Search time
(f) Verification time
GString Recall, Accuracy

- **Recall**
  - Fraction of relevant instances that are retrieved

- **Precision (Accuracy)**
  - Fraction of retrieved instances that are relevant
(a) Recall
(b) Accuracy
Comparison, results

- The authors have compared their approach against GraphGrep, Gindex, and C-tree
  - default / recommended parameters

- For subgraph queries “approach outperforms GraphGrep and GIndex”
  - Index size & construction time,
  - candidate set size
  - Subgraph search accuracy, efficiency
Comparison (cont.)

- Compared to C-tree
  - “our approach is advantageous in all measures except for index construction time”
Conclusion

- **GString**
  - String conversion for a specific type of graph
  - Then use string indexing techniques
    - Support for subgraph queries
    - Support for similarity search
- **Compact indexing structures**
- **Accuracy**
- **Efficiency**
- … final verdict? …