Kernel-based similarity search in massive graph databases with wavelet trees

Yasuo Tabei and Koji Tsuda

JST ERATO Minato Project, Japan
AIST Comp Bio Research Center, Japan
Graph similarity search

- Similarity search for 25 million molecular graphs
  - Find all graphs whose similarity to the query $\geq 1 - \epsilon$
  - Similarity = Weisfeiler-Lehman graph kernel (NIPS, 2009)

- Use data structure called “Wavelet Tree” (SODA, 2003)
  - Self-index of an integer array
    - Combination of rank dictionaries of bit arrays
  - Enables fast array operations
    - e.g., range minimum query, range intersection

- First review wavelet tree in the range intersection problem, then graph similarity search
Range intersection on array

- Array $A$ of length $N$, $1 \leq A_i \leq M$

  ![](image)

- Range intersection: $\text{rint}(A, [i,j], [k,\ell])$
  - Find set intersection of $A[i,j]$ and $A[k,\ell]$
  - The naïve solution is to concatenate and sort

- Use an index structure of the array (Wavelet Tree) and solve the problem faster!

11/05/09
Tree of subarrays:
Lower half = left, Higher half=right

[1,8]
[1,4] [5,8]
[1,2] [3,4] [5,6] [7,8]
Remember if each element is either in lower half (0) or higher half (1)

[1,8]

[1,4]

[1,2]

[3,4] [5,6]

[7,8]
Index each bit array with a rank dictionary

- With rank dictionary, the rank operation can be done in O(1) time
  - \( \text{rank}_c(B,i) \): return the number of \( c \in \{0,1\} \) in \( B[1 \ldots i] \)
- Several algorithms known: \( \text{rank9sel} \)

Example) \( B=0110011100 \)

\[
\begin{array}{cccccccccccc}
  i & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
  B & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
  \text{rank}_1(B,8) & 5 & & & & & & & & & \\
  \text{rank}_0(B,5) & 3 & & & & & & & & & \\
\end{array}
\]
Implementation of rank dictionary

- Divide the bit array $B$ into large blocks of length $l = \log^2 n$
  - $R_L$ = Ranks of large blocks
- Divide each large block to small blocks of length $s = \log n / 2$
  - $R_s$ = Ranks of small blocks relative to the large block

$$\text{rank}_1(B,i) = R_L[i/l] + R_s[i/s] + \text{(remaining rank)}$$

Time: $O(1)$
Memory: $n + o(n)$ bits
O(1)-time division of an interval

- Using the rank operations, the division of an interval can be done in \textit{constant time}.
  - \( \text{rank}_0 \) for left child and \( \text{rank}_1 \) for right child.

- Naïve = \textit{linear time} to the total number of elements.
Fast computation of rank intersection by pruning

Pruned solution!!
Graph Similarity Search

- Bag-of-words representation of graph
  - Weisfeiler-Lehman procedure (NIPS, 2009), Hido and Kashima (2009), Wang et al., (2009)

- Cosine similarity query
  - Find all graphs $W$ whose cosine similarity (kernel) to the query $Q$ is at least $1-\varepsilon$

$$K_N(W, Q) = \frac{|W \cap Q|}{\sqrt{|W||Q|}}.$$
Weisfeiler-Lehman Procedure *(NIPS,09)*

- Convert a graph into a set of words (bag-of-words)

  i) Make a label set of adjacent vertices  
     ex) \{E,A,D\}

  ii) Sort  
     ex) A,D,E

  iii) Add the vertex label as a prefix  
     ex) B,A,D,E

  iv) Map the label sequence to a unique value  
     ex) B,A,D,E→R

  v) Assign the value as the new vertex label

Bag-of-words \{A,B,D,E,…,R,…\}
Semi-conjunctive query

- Cosine similarity query can be relaxed to the following form

\[ W \text{ s.t. } |W \cap Q| \geq k \]

- Find all graphs \( W \) which share at least \( k \) words to the query \( Q \)
- No false negatives
- False positives can easily be filtered out by cosine calculations
Inverted index, Array, Wavelet Tree

- Graph database is represented as inverted index
- Concatenate all rows to make an array
- Index the array with wavelet tree
- Semi-conjunctive query = Extension of range intersection
  - Find graph ids which appear at least k times in the array
Pruning condition

- Find all graphs $W$ in the database whose cosine is larger than a threshold $1-\varepsilon$

$$W \ s.t \ K_N(W, Q) = \frac{|W \cap Q|}{\sqrt{|W|} \sqrt{|Q|}} \geq 1 - \varepsilon$$

- $W, Q$: bag-of-words of graphs

- The above solution can be relaxed as follows,

If $$K_N(Q, W) \geq 1 - \varepsilon$$, then

$$(1-\varepsilon)^2 |Q| \leq |W| \leq \frac{|Q|}{(1-\varepsilon)^2}$$

- Can be used for pruning
Complexity

- Time per query: $O(\tau m)$
  - $\tau$: the number of traversed nodes
  - $m$: the number of bag-of-words in a query
- Memory: $(1+\alpha)N \log n + M \log N$
  - $N$: the number of all words in the database
  - $M$: Maximum integer in the array
  - $n$: the number of graphs
  - $\alpha$: overhead for rank dictionary ($\alpha=0.6$)
- Inverted index takes $N \log n$ bits
- About 60% overhead to inverted index!
Experiments

- 25 million chemical compounds from PubChem database
- Evaluated search time and memory usage
- Cosine thresholds $\varepsilon=0.3,0.35,0.4$
- Compare our method $gWT$ to
  - Inverted index (concatenate all intervals and sort)
  - Sequential scan (compute kernel one by one)
Search time

![Graph showing search time vs. number of graphs]

- gWT 0.4
- gWT 0.35
- gWT 0.3
- Inverted index
- Seq. Scan

Search times:
- 40 sec
- 38 sec
- 8 sec
- 3 sec
- 2 sec
Memory usage

![Graph showing memory usage]
Construction time

- bag-of-words
- wavelet tree

$\leftarrow 7h$
Related work

- A lot of methods have been proposed so far.
  1. Graph Grep [Shasha et al., 02]
  2. gIndex [Yan et al., 04]
  3. Tree+Delta [Zhao et al., 07]
  4. TreePi [Zhang et al., 07]
  5. Gstring [Jiang et al., 07]
  6. FG-Index [Cheng et al., 07]
  7. GDIndex [Williams et al., 07]
  etc
Related work

- Decompose graphs into a set of substructures
  - graphs, trees, paths etc
- Build a substructure based index
- Drawbacks
  - Need substructure mining
  - Do not scale to millions of graphs
Summary

- Efficient similarity search method of massive graph databases
- Solve semi-conjunctive queries efficiently
- Our method is built on wavelet trees
- Use Weisfeiler-Lehman procedure to convert graphs into bag-of-words
- Applicable to 25 million graphs

Software

http://code.google.com/p/gwt/