Query Preserving Graph Compression

Wenfei Fan 1,2, Jianzhong Li 2, Xin Wang 1, Yinghui Wu 1,3
1University of Edinburgh 2Harbin Institute of Technology 3UC Santa Barbara
{wenfei@inf..x.wang-36@sms.,y.wu-18@sms}ed.ac.uk, lijzh@hit.edu.cn

Introduction
- Queries over large real-life graphs are prohibitively expensive.
- Reachability queries: \( O(|V|+|E|) \) for \( G(V,E) \)
- Incremental methods with construction and maintenance cost
- Likely to lower the computational complexity
- Graph compression: construct compressed graphs which preserve information only related to a class of queries of users' choice

Querying Recommendation Network

Query Preserving Graph Compression

Query Preserving Graph Compression. A triple \(<R,F,P>\) where
- \(R\): a compression function,
- \(P\subseteq L_o \times L_o\), a query rewriting function for a class of graph queries \(L_o\), and
- \(P\): a post-processing function.

Query Preserving Graph Compression

Graph Pattern Preserving Compression

Bisimulation relation. A binary relation \(B\) over \(V\) of \(G\), s.t., for each \((u,v)\in B\),
- the label of \(u\) and \(v\) are equivalent, and
- for each \(u\)'s (resp. \(v\)'s) child \(u'\) (resp. \(v'\)), \(v\) (resp. \(u\)) has a child \(v'\) (resp. \(u'\)) that \((u',v')\in B\).

Theorem: There is a graph pattern preserving compression \(<R,F,P>\) for \(G\) where
- \(R\) maps each node \(v\) in \(G\) to its bisimulation equivalence class \([v]\in \text{O}(\text{E}(\log |V|))\)
- \(F\) is the identity mapping
- \(P\) maps each query node \(u\) and its match (as an equivalence class \([u]\)) to node pairs \((u,v')\) for each \(v'\in [u]\) (linear time in the size of query result)

Algorithm
- Compute the unique maximum bisimulation relation by iteratively refine the equivalence classes (initialized as \(V\)).
- Construct the compression graph \(G\), where each node denotes a bisimulation equivalence class, and each edge connects two nodes \([v_1]\) and \([v_2]\) if \((v_1,v_2)\) is an edge in \(G\).

Incremental Query Preserving Compression

Real-life graphs are changing. To compute the compressed graph from scratch is expensive.
- Incremental graph compression: given a data graph \(G\), its changes \(\Delta G\), and a compressed graph \(G_p\), compute \(\Delta G_p\), i.e., changes to \(G_p\), such that \(G_p \oplus \Delta G_p = G_p \oplus \Delta G\)

Affected area: the total changes in the data graph \(\Delta G\) and the compressed graph \(\Delta G_p\). Unbounded, bounded, optimal ...

Incremental reachability preserving compression is unbounded even for unit updates, and is in \(O(|AFQ(G)|)\) time
Incremental pattern preserving compression is unbounded for unit updates, and is in \(O(|AFQ(G)|)\) time

Algorithms
- Update the ranks of the nodes (blocks), identify initial affected area
- Split-merge the blocks and propagate the affected area, until a fixpoint is reached

Incremental maintenance
- Construct compressed graphs that can be directly queried without decompression
- Reachability and pattern preserving compression are efficient, and can be maintained without accessing original graphs

Experimental Study

Compressing P2P network

Compression Ratio: Reachability (in average 5%) and Pattern Preserving (in average 45%)

Query efficiency: Reachability (in average 2%) and Pattern Preserving (in average 30%)

Conclusion