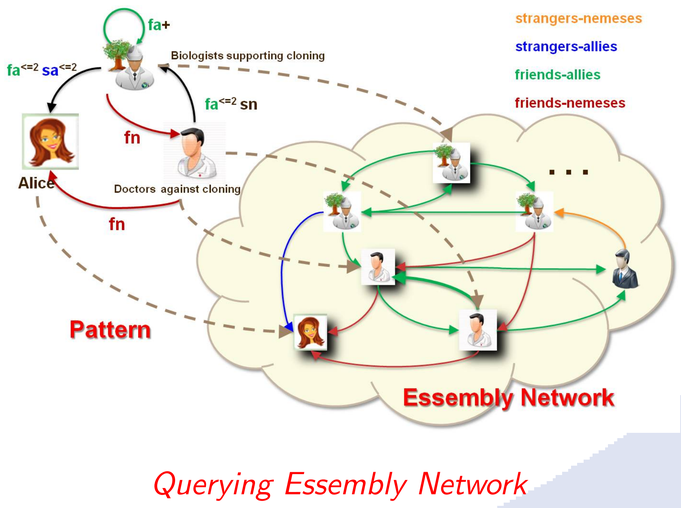
Adding Regular Expressions to Graph Reachability and Pattern Queries

Wenfei Fan [1,2], *Jianzhong Li* [2], *Shuai Ma* [1], *Nan Tang* [1], *Yinghui Wu* [1]
[1] *University of Edinburgh* [2] *Harbin Institute of Technology*

{wenfei@inf.,shuai.ma@,ntang@inf.,y.wu-18@sms.}ed.ac.uk,lijzh@hit.edu.cn

Introduction

- Real-life graphs bear different edge types, indicating a variety of relationships.
- Various graph querying semantics:
  - subgraph isomorphism: **function-based, edge-edge matching**.
  - graph simulation: **relation-based, edge-edge matching**.
- Existing methods cannot meet the demands in emerging applications, *e.g.*, social network.
- We revise graph simulation by adding *regular expressions*, to find more sensible information than their traditional counterparts.



Querying Essembly Network

Graph reachability and pattern queries

Data Graph. A directed graph with node properties (*e.g.,* labels, keywords, blogs, comments, ratings) and edge types (*e.g.,* marriage, friendship, work, co-membership).

Reachability queries. A *reachability query* (RQ) is in the form of an edge (u_1, u_2), where

- u_1 and u_2 are two nodes,
- each node carries a predicate as a conjunction of atomic formulas (*e.g.,* job='doctor'),
- an RQ bears a regular expression drawn from the subclass:

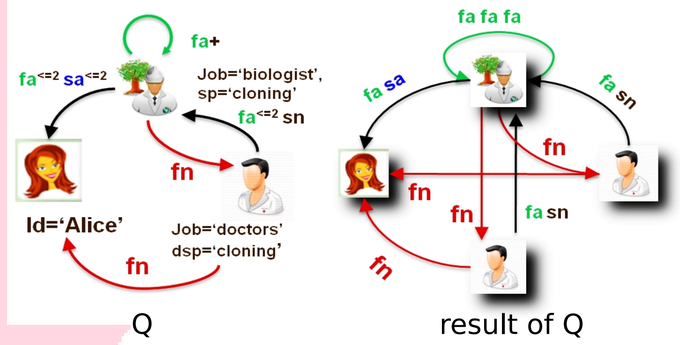
$$F ::= \ c \mid c^{\leq k} \mid c^{+} \mid FF.$$

(c represents a colour, k is a positive integer, and c^+ denotes one or more occurrences of c),

- query result is a set of node pairs, each pair is connected by a path satisfying the length and colour constraints of RQ.

A reachability query and its result graph

Graph pattern queries. A *graph pattern query* (PQ) is a directed graph where each edge is an RQ. The result of PQ is the recursively defined union of the results for its edges. The result for an edge in PQ is the result of RQ the edge represents.



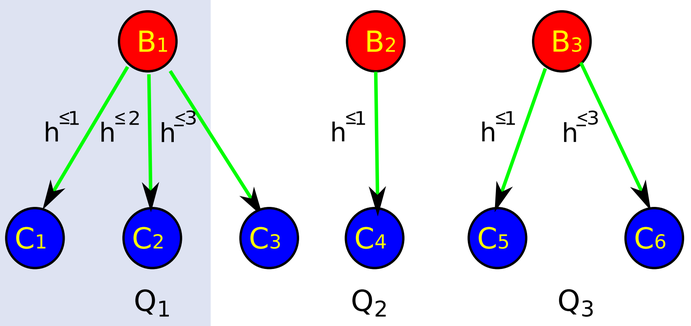
A graph pattern query and its result graph

Fundamental problems

Containment. Given two PQs Q_1 and Q_2, Q_1 is *contained* in Q_2, if for *all* data graph, the result of each edge in Q_1 is contained in the result of an edge in Q_2.

Equivalence. Two PQs Q_1 and Q_2 are *equivalent*, iff they are *contained* in each other.

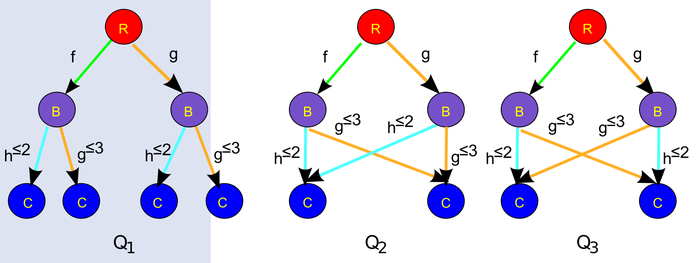
Theorem: Given two PQs Q_1 and Q_2, it is in cubic time to determine whether Q_1 is contained in, or equivalent to Q_2.



Query Containment and Equivalence

Query minimization. The *minimization* problem is to find, for a given PQ Q, another PQ Q_m that is equivalent to Q and has a minimum size (the sum of nodes and edges).

Theorem. Given any PQ Q, a minimum equivalent PQ Q_m of Q can be computed in cubic time.



Query Minimization

Algorithms

Reachability queries

An RQ query can be evaluated in quadratic time, by capitalizing a matrix of shortest distances.

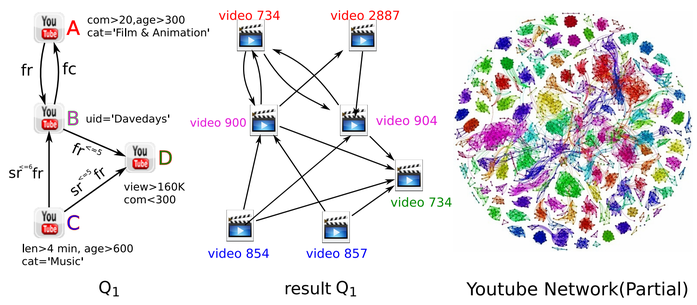
Graph pattern queries

Given a PQ Q and a data graph G, Q can be evaluated in cubic time.

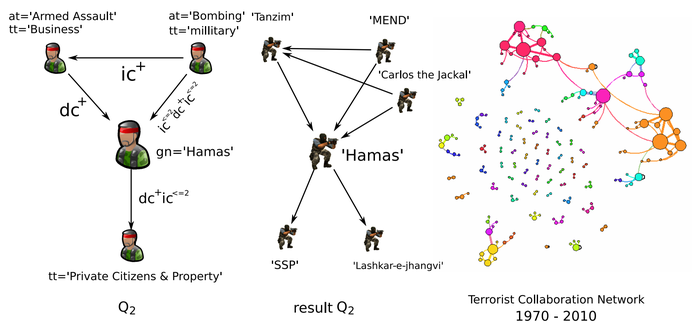
Two cubic-time algorithms.

- *Join-based* algorithm:
  - Initialize candidates for query nodes.
  - Join operation for query edges till fixpoint.
- *Split-based* algorithm:
  - Initialize over-estimated partition-relation pair for query nodes.
  - Split blocks and filter candidates till fixpoint.

Experimental results



Querying Youtube Network



Querying Terrorist Network

Summary:

- PQs are able to identify far more sensible matches in emerging application than the conventional approaches.
- PQs can be efficiently evaluated, and scale well with large graphs and complex patterns.

Conclusion

- Extensions of reachability queries (RQs) and graph pattern queries (PQs) by incorporating a subclass of regular expressions to capture edge relationships .
- Fundamental problems (containment, equivalence, minimization) for these queries are all in low PTIME.
- Two *cubic-time* algorithms for evaluating PQs.

Future work

- Extend RQs and PQs by supporting general regular expressions.
- Identify application domains in which simulation-based PQs are most effective.
- Find incremental evaluation algorithms that guarantee to minimize unnecessary recomputation.

School of informatics