

# Large Scale Graph Algorithms

A Guide to Web Research: Lecture 2

Yury Lifshits

Steklov Institute of Mathematics at St.Petersburg

Stuttgart, Spring 2007

To pose an abstract computational problem on graphs that has a huge list of applications in web technologies

## Outline

- 1 Family of Problems: Finding Strongest Connection
  - Problem Statement and Applications
  - Variations of Strongest Connection Problem
- 2 Max-Intersection Problem
  - Statement and Naive Solutions
  - Hierarchical Schema Solution
- 3 Concluding Remarks
  - Overview of Related Research
  - Open Problems

## Part I

### Family of Problems: Finding Strongest Connection

Problem statement  
Applications  
Variations of the problem

## Strongest Connection Problem (SCP)

**BASIC SETTINGS:** a class of graphs  $\mathcal{G}$ , a class of paths  $\mathcal{P}$

**INPUT:** a graph  $G \in \mathcal{G}$

Allowed time for preprocessing:  $o(|G|^2)$

**QUERY:** a (new) vertex  $v$

**TASK:** to find a vertex  $u \in G$

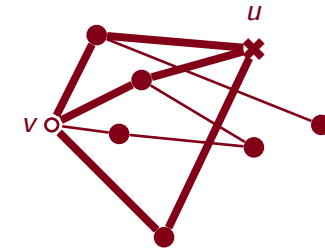
that has maximal number of  $\mathcal{P}$ -paths from  $v$  to  $u$

Allowed time for query processing:  $o(|G|)$

5 / 34

## Homogeneous Graph / 2-Step Paths

Graph of coauthoring



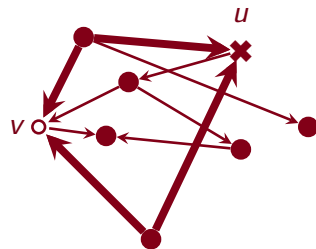
Coauthor suggest in **DBLP**

The most common coauthor of my coauthors

6 / 34

## Directed Graph / 2-Step Paths

Graph of hyperlinks



Advanced option for **Google search**: link-based similar website

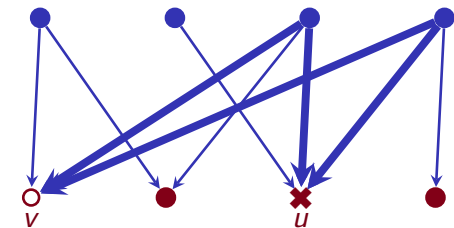
The website that is most often co-cited with the given one

7 / 34

## Bipartite Graph / 2-Step Paths

People

Bands



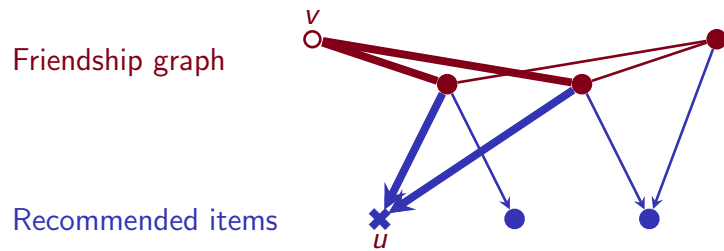
**Last.fm** similar music bands

The band that is most often co-listened with the given one

In general: any content-based similarity, keyword-similarity, any co-occurrence similarity

8 / 34

## Homogeneous-Bipartite Graph / 2-Step Paths

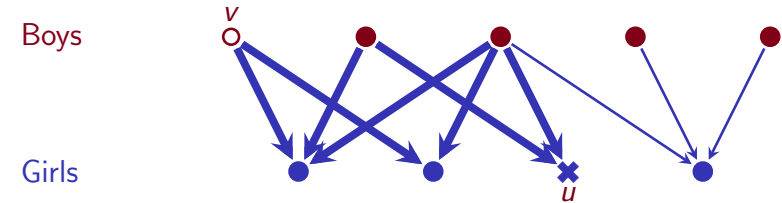


Social recommendations in networks like **Facebook**  
System recommends things that are popular among my friends

9 / 34

## Bipartite Graph / 3-Step Paths

New girlfriend suggest:

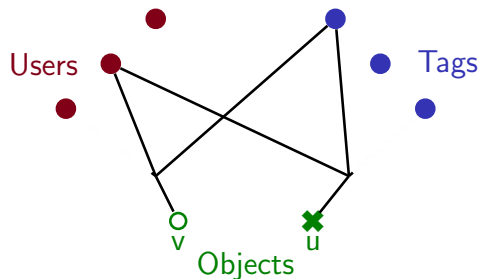


**Amazon.com** recommendations  
Subscription recommendations for **FeedBurner**, **Google Reader**  
Items that have the largest number of co-occurrences with my items

10 / 34

## Tripartite 3-Graph / 2-Step Paths

**Folksonomy** is a set of triples  $\langle user, tag, object \rangle$



Similar websites in **Del.icio.us**, similar pictures in **Flickr**  
Largest number of common tags  
Largest number of common users  
Largest number of common pairs  $\langle user, tag \rangle$

11 / 34

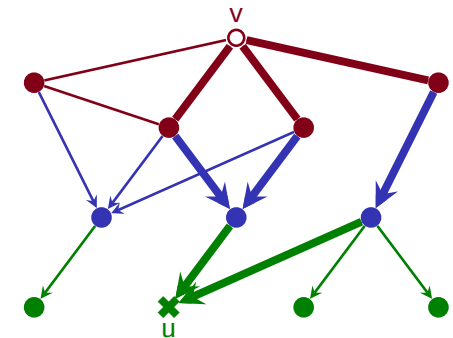
## Multicolor-Multiparty Graph / k-Step Paths

Semantic search: "Most popular drink that is available on bars that are visited by my friends"

Friendship graph

Bar visiting

Drinks in menu



12 / 34

## Variations of Strongest Connection Problem

- Directed/undirected graphs
- Weights on edges/vertices
- Task: offline, on-line, all-to-all
- Task: one best connection,  $k$  best connections
- Graph and weights are evolving with time

13 / 34

## Solution Variations

### Usual alternatives to exact algorithm:

- Approximate algorithms
- Randomized algorithms
- Input graph (or query) belongs to a certain distribution. Average complexity analysis
- Introducing additional assumptions
- Introducing additional input-complexity parameter
- Modifying the computation task
- Heuristics
- Look to particular cases (subproblems)

15 / 34

## Claim

Computing strongest connection is probably the most important algorithmic problem related to web technologies★

★Personal opinion of Yury Lifshits

14 / 34

## Part II Max-Intersection Problem

Statement and naive solutions  
Hierarchical schema solution

This section represents a work-in-progress joint research with Benjamin Hoffmann and Dirk Nowotka

16 / 34

## Statement of Max-Intersection Problem

### In set notation:

Input: Family  $\mathcal{F}$  of  $n$  sets,  $\forall f \in \mathcal{F} \quad |f| \leq k$

Time for preprocessing:  $n \cdot \text{polylog}(n) \cdot \text{poly}(k)$

Query: a set  $f_{new}$ ,  $|f_{new}| \leq k$

Task: Find  $f_i \in \mathcal{F}$  that maximizes  $|f_{new} \cap f_i|$

Time for query processing:  $\text{polylog}(n) \cdot \text{poly}(k)$  or at most  $o(n)$



### In bipartite graph notation:

Input: Bipartite documents-terms graph,  $|\mathcal{D}| = n$ ,  $\forall d \in \mathcal{D} \quad |d| \leq k$

Query: a document  $d_{new}$ ,  $|d_{new}| \leq k$

Task: Find  $d_i \in \mathcal{D}$  that has maximal number of common terms with  $d_{new}$

17 / 34

## Applications of Max-Intersection (1/2)

### Homogeneous graphs:

- **References** in scientific papers: (1) maximal number of co-occurrences in reference list (2) maximal intersection of reference lists
- **Social networks** (e.g. LinkedIn): a person that has maximal connections with my direct neighborhood
- **Collaboration networks** (e.g. DBLP): given a scientist, to find another one with maximal overlapping of coauthors-list

### Bipartite graphs:

- **Websites—Words** graph: find a website with maximal intersection of used terms with the given one
- **Music\_Bands—Listeners** graph: find a band that has maximal intersection of listeners with the given one

18 / 34

## Applications of Max-Intersection (2/2)

### Tripartite graphs:

- **Long\_Search\_Queries—Web\_Dictionary—Websites**: given a query to find a website with maximal number of query terms
- **Advertisement\_Description—Keywords—Websites** (e.g. AdSense Matching): find a website with maximal number of terms from advertisement description
- **PC\_Members—Keywords—Submissions**: find a paper that has maximal number of terms that belong to expertise of the given PC member

19 / 34

## Inverted Index (1/2)

Let us use documents-terms notation

### Inverted index approach:

- Preprocessing. For every term produce a list of all documents that contain it  
Complexity:  $O(n \cdot k)$
- Query  $d_{new} = \{t_1, \dots, t_k\}$ . Retrieve document lists for all terms of query. Check all documents in all these  $k$  lists and return the one with maximal intersection with  $d_{new}$   
Worst case complexity:  $\Omega(n)$

Let  $T_{max}$  be the maximal degree of terms. Then the query complexity is  $O(k \cdot T_{max})$

20 / 34



### Theorem

*With very high probability there are no  $d \in \mathcal{D}$  that has at least  $q' + \varepsilon$  common elements with  $d_{new}$*

25 / 34

## Part III Concluding Remarks

Overview of related research  
**Open problems**

27 / 34

### Preprocessing:

Encode every document as a  $2k - 1$  sequence, every odd element lies in range  $[1..k]$ , every even is 0 or 1  
Construct a lexicographic tree for all encodings

### Query processing:

Find the largest **prefix-match** between  $d_{new}$  and documents from  $\mathcal{D}$

By two theorems above with very high probability maximal prefix-match is very close to maximal intersection

26 / 34

## Overview of Related Research

Famous computational problems that need scalable algorithms:

- Nearest neighbors in vector spaces
- Nearest neighbors in abstract metric spaces
- Connection subgraph problem
- Collaborative filtering
- Mining association rules
- Indexing with errors

Common approach: heuristical algorithm + experimental validation

**Alternative:** randomized model of input + probabilistic analysis

**Alternative:** realistic assumption about input + exact algorithm

28 / 34

## Algorithms for Max-Intersection

### Algorithmic open problems:

- 1 Max-Intersection for bounded tree-width graphs
- 2 Max-Intersection in configuration model
- 3 Max-Intersection in preferential attachment model

### Conceptual open problem:

- 1 Find simple-but-realistic assumptions allowing required exact solution of Max-Intersection

**Long-term goal:** to develop **theoretical** framework for scalability analysis of algorithms

29 / 34

## Call for participation

Know a relevant reference?

Have an idea?

Find a mistake?

Solved one of these problems?

- Knock to my office 1.156
- Write to me yura@logic.pdmi.ras.ru
- Join our informal discussions
- Participate in writing a follow-up paper

31 / 34

## Data Structure Complexity

On-line inclusion problem

**Input:** Family  $\mathcal{F}$  of  $2^k$  subsets of  $[1..k^2]$

Data storage after preprocessing:  $2^k \cdot \text{poly}(k)$

**Query:** a set  $f_{\text{new}} \subseteq [1..k^2]$

**Task:** decide whether  $\exists f \in \mathcal{F} : f_{\text{new}} \subseteq f$

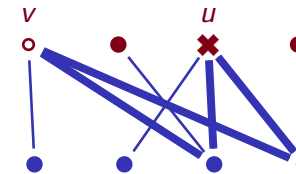
Time for query processing:  $\text{poly}(k)$

**Conjecture:** the on-line inclusion problem **can not** be solved within such time/space constraints

30 / 34

## Highlights

**Strongest Connection** family, including **Max-Intersection**



**Open problems:**

Max-Intersection in **complex-networks models**

Data structure complexity of **on-line inclusion problem**


Vielen Dank für Ihre Aufmerksamkeit! Fragen?

32 / 34






## References (1/2)

**Course homepage** <http://logic.pdmi.ras.ru/~yura/webguide.html>

-  Y. Lifshits  
Web research: open problems  
<http://logic.pdmi.ras.ru/~yura/en/web-talk.pdf>
-  J. Zobel and A. Moffat  
Inverted files for text search engines  
<http://portal.acm.org/citation.cfm?id=1132959>
-  C. Faloutsos, K.S. McCurley, A. Tomkins  
Fast discovery of connection subgraphs  
<http://www.cs.cmu.edu/~christos/PUBLICATIONS/kdd04-conn-graphs.pdf>
-  M.E.J. Newman  
The structure and function of complex networks  
<http://arxiv.org/abs/cond-mat/0303516,2003>

## References (2/2)

-  P.N. Yianilos  
Data structures and algorithms for nearest neighbor search in general metric spaces  
<http://www.pnylab.com/pny/papers/vptree/vptree.ps>
-  J. Kleinberg  
Two algorithms for nearest-neighbor search in high dimensions  
<http://www.ece.tuc.gr/~vsam/csalgo/kleinberg-stoc97-nn.ps>
-  R. Agrawal and R. Srikant  
Fast algorithms for mining association rules in large databases  
<http://www.cs.indiana.edu/hyplan/dgroth/P487.PDF>
-  M. O'Connors J. Herlocker  
Clustering items for collaborative filtering  
<http://www.cs.umbc.edu/~ian/sigir99-rec/papers/oconner.m.pdf>