Large Scale Graph Algorithms

A Guide to Web Research: Lecture 2

Yury Lifshits

Steklov Institute of Mathematics at St.Petersburg

Stuttgart, Spring 2007

Talk Objective

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

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

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)

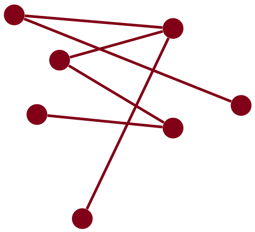
BASIC SETTINGS: a class of graphs $\mathcal{G}$, a class of paths $\mathcal{P}$

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: $o(|G|)$

Homogeneous Graph / 2-Step Paths

Graph of coauthoring

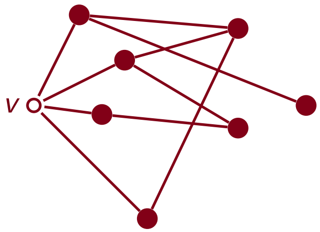


Coauthor suggest in **DBLP**

The most common coauthor of my coauthors

Homogeneous Graph / 2-Step Paths

Graph of coauthoring

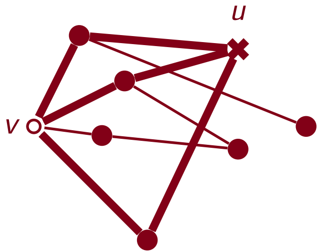


Coauthor suggest in **DBLP**

The most common coauthor of my coauthors

Homogeneous Graph / 2-Step Paths

Graph of coauthoring

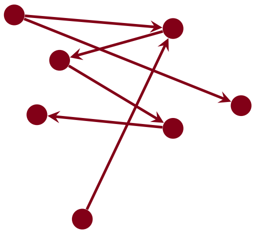


Coauthor suggest in **DBLP**

The most common coauthor of my coauthors

Directed Graph / 2-Step Paths

Graph of hyperlinks

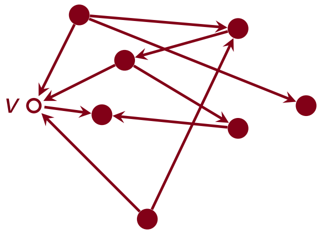


Advanced option for **Google search**: link-based similar website

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

Directed Graph / 2-Step Paths

Graph of hyperlinks

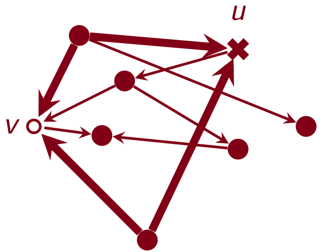


Advanced option for **Google search**: link-based similar website

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

Directed Graph / 2-Step Paths

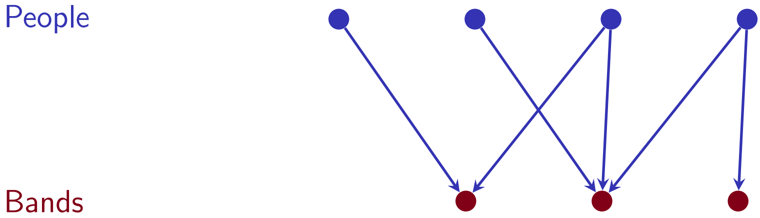
Graph of hyperlinks



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

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

Bipartite Graph / 2-Step Paths

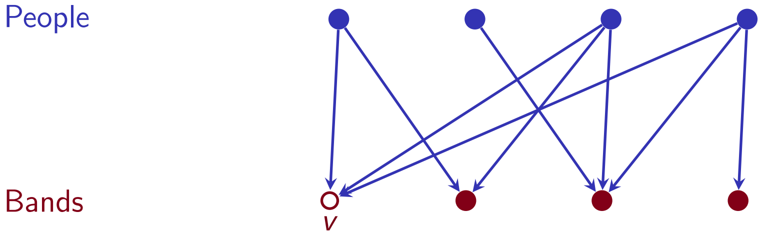


Last.fm similar music bands

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

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

Bipartite Graph / 2-Step Paths

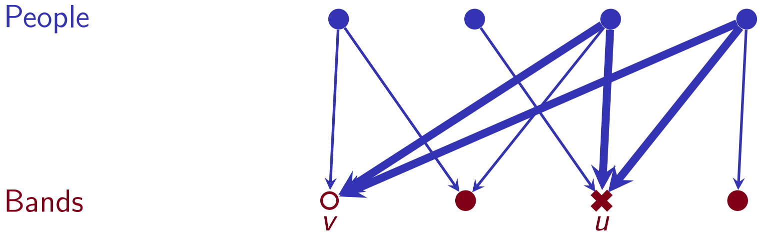


Last.fm similar music bands

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

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

Bipartite Graph / 2-Step Paths



Last.fm similar music bands

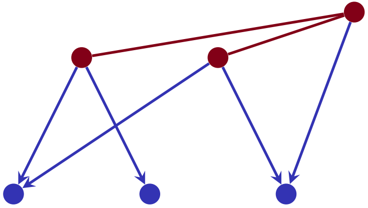
The band that is most often co-listened with the given one

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

Homogeneous-Bipartite Graph / 2-Step Paths

Friendship graph

Recommended items



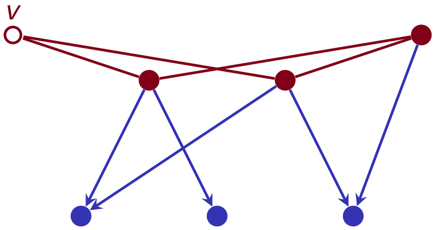
Social recommendations in networks like **Facebook**

System recommends things that are popular among my friends

Homogeneous-Bipartite Graph / 2-Step Paths

Friendship graph

Recommended items



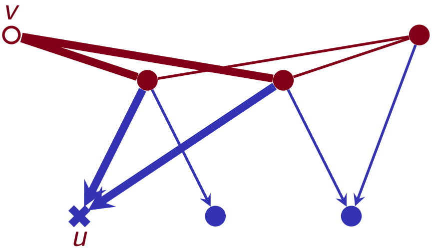
Social recommendations in networks like **Facebook**

System recommends things that are popular among my friends

Homogeneous-Bipartite Graph / 2-Step Paths

Friendship graph

Recommended items

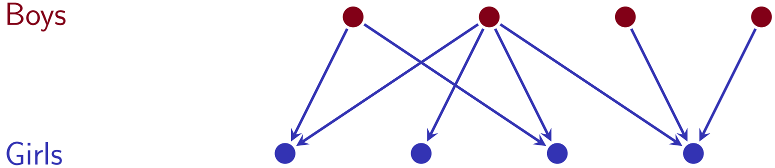


Social recommendations in networks like **Facebook**

System recommends things that are popular among my friends

Bipartite Graph / 3-Step Paths

New girlfriend suggest:



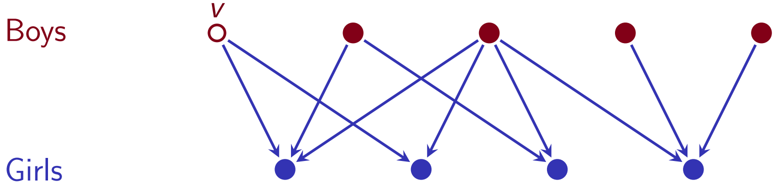
Amazon.com recommendations

Subscription recommendations for **FeedBurner**, **Google Reader**

Items that have the largest number of co-occurrences with my items

Bipartite Graph / 3-Step Paths

New girlfriend suggest:



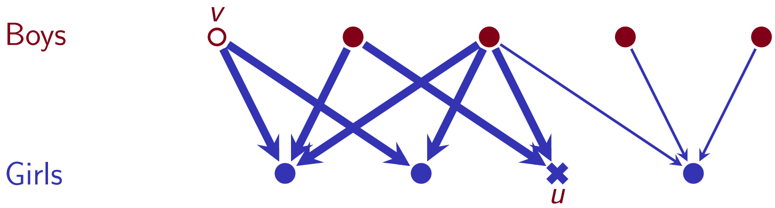
Amazon.com recommendations

Subscription recommendations for **FeedBurner**, **Google Reader**

Items that have the largest number of co-occurrences with my items

Bipartite Graph / 3-Step Paths

New girlfriend suggest:



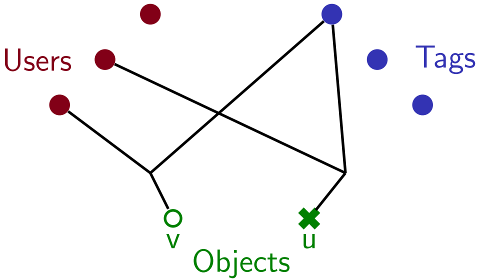
Amazon.com recommendations

Subscription recommendations for **FeedBurner**, **Google Reader**

Items that have the largest number of co-occurrences with my items

Tripartite 3-Graph / 2-Step Paths

Folksonomy is a set of triples $< user, tag, object >$

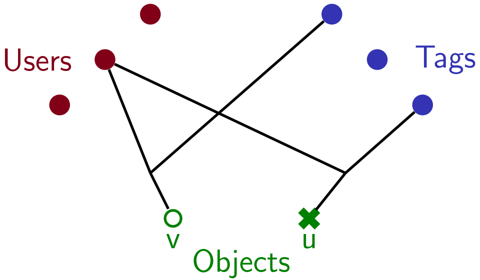


Similar websites in **Del.icio.us**, similar pictures in **Flicr**

Largest number of common tags
Largest number of common users
Largest number of common pairs $< user, tag >$

Tripartite 3-Graph / 2-Step Paths

Folksonomy is a set of triples $< user, tag, object >$



Similar websites in **Del.icio.us**, similar pictures in **Flicr**

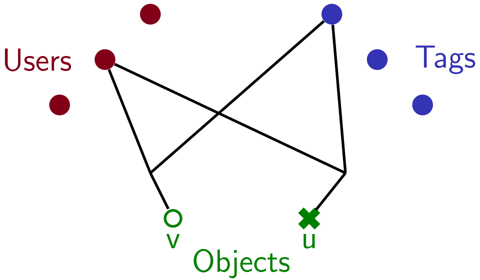
Largest number of common tags

Largest number of common users

Largest number of common pairs $< user, tag >$

Tripartite 3-Graph / 2-Step Paths

Folksonomy is a set of triples $< user, tag, object >$



Similar websites in **Del.icio.us**, similar pictures in **Flicr**

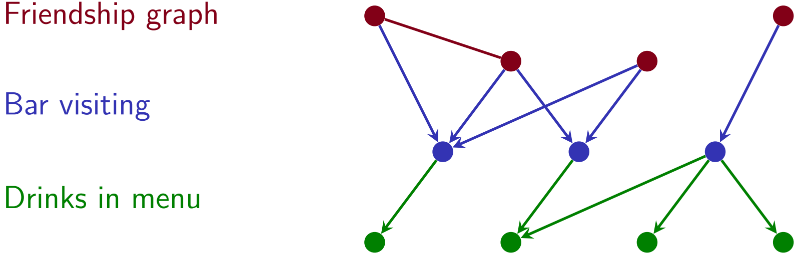
Largest number of common tags

Largest number of common users

Largest number of common pairs $< user, tag >$

Multicolor-Multiparty Graph / k-Step Paths

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



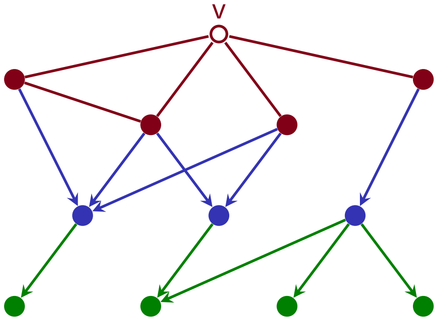
Multicolor-Multiparty Graph / k-Step Paths

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

Friendship graph

Bar visiting

Drinks in menu



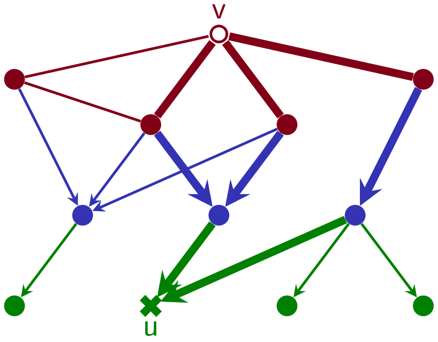
Multicolor-Multiparty Graph / k-Step Paths

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

Friendship graph

Bar visiting

Drinks in menu



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

Claim

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

Claim

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

★Personal opinion of Yury Lifshits

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)

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

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: $n \cdot polylog(n) \cdot poly(k)$

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: $polylog(n) \cdot poly(k)$

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: $n \cdot polylog(n) \cdot poly(k)$

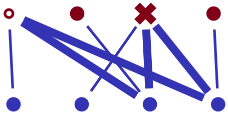
Query: a set f_{new}, $|f_{new}| \leq k$
Task: Find $f_i \in \mathcal{F}$ that maximizes $|f_{new} \cap f_i|$
Time for query processing: $polylog(n) \cdot poly(k)$ or at most $o(n)$

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: $n \cdot polylog(n) \cdot poly(k)$

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: $polylog(n) \cdot poly(k)$ 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}

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

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

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 form advertisement description
- PC_Members—Keywords—Submissions: find a paper that has maximal number of terms that belong to expertise of the given PC member

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 $d_{new} = \{t_1, \ldots, t_k\}$. 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: $\Omega(n)$

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 $d_{new} = \{t_1, \ldots, t_k\}$. 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 $O(k \cdot T_{max})$

Inverted Index (2/2) Rare-Term Requirement

Cheating: modify the Max-Intersection problem

New Task: Given the document d_{new}, find a document d_i such that

1. It has a joint **rare term** (term that occurs in at most r documents) with d_{new}
2. The intersection with d_{new} is maximal among all documents satisfying (1)

Inverted Index (2/2) Rare-Term Requirement

Cheating: modify the Max-Intersection problem

New Task: Given the document d_{new}, find a document d_i such that

1. It has a joint **rare term** (term that occurs in at most r documents) with d_{new}
2. The intersection with d_{new} is maximal among all documents satisfying (1)

Observation

Inverted index can handle queries in $O(r \cdot k)$ time now

Inverted Set-Index

Assume that k is extremely small, say $k = O(\log\log n)$

Inverted set-index approach:

- Preprocessing. Write down all term subsets of all documents. Sort all these subsets in lexicographical order

  Complexity: $O(n \cdot 2^k)$

- Query $d_{new} = \{t_1, \ldots, t_k\}$. For every subset of query terms search it in the inverted set-index. Return the document that corresponds to the maximal subset founded in index

  Complexity: $O(2^k(k + \log n))$

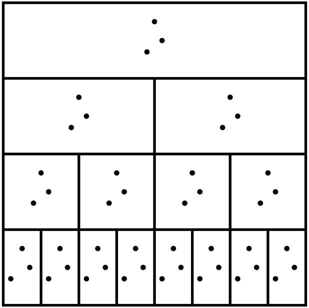
Hierarchical Schema

Table of terms:

k levels
Level i is divided to 2^{i-1} cells
Every cell contains k terms

Random nature of $\mathcal{D}$ and d_{new}:

Choose random cell on the bottom level
Mark all cells that are above it
Choose one random term in every marked cell



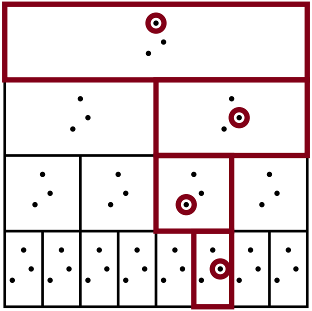
Hierarchical Schema

Table of terms:

k levels
Level i is divided
to 2^{i-1} cells
Every cell
contains k terms

Random nature of $\mathcal{D}$ and d_{new}:

Choose random cell
on the bottom level
Mark all cells that are above it
Choose one random term
in every marked cell



Magic Levels (1/2)

Assume that there are 2^k such "random" documents in $\mathcal{D}$

Notation: **magic levels** $q = \frac{k}{\log k+1}, \quad q' = \frac{k}{\log k}$

Theorem

With very high probability there exists $\mathbf{d} \in \mathcal{D}$ that has the same terms from top $\mathbf{q} - \varepsilon$ levels

Magic Levels (2/2)

Theorem

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

Algorithm for Hierarchical Schema

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 $\mathcal{D}$

Algorithm for Hierarchical Schema

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

Part III
Concluding Remarks

Overview of related research

Open problems

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

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

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

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

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

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

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

Data Structure Complexity

On-line inclusion problem

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

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

Query: a set $f_{new} \subseteq [1..k^2]$
Task: decide whether $\exists f \in \mathcal{F} : \quad f_{new} \subseteq f$

Time for query processing: $poly(k)$

Data Structure Complexity

On-line inclusion problem

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

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

Query: a set $f_{new} \subseteq [1..k^2]$
Task: decide whether $\exists f \in \mathcal{F}: \quad f_{new} \subseteq f$

Time for query processing: $poly(k)$

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

Call for participation

Know a relevant reference?

Have an idea?

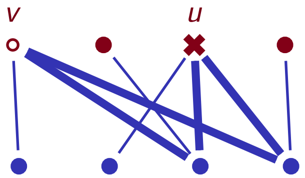
Find a mistake?

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

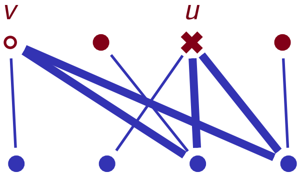
Highlights

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



Highlights

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



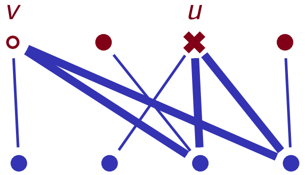
Open problems:

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

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

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?

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