Aalto University
School of Science

Algorithmic methods for mining large graphs
Lecure 2 : Computing basic graph statistics

Aristides Gionis

Aalto University

Bertinoro International Spring School 2016
March 7–11, 2016

course agenda

• introduction to graph mining	Tue afternoon
• computing basic graph statistics	Tue afternoon, Wed morning
• finding dense subgraphs	Wed afternoon, Thu morning
• spectral graph analysis	Thu afternoon
• additional topics	Fri morning

– inferring hierarchies in graphs

– mining dynamic graphs

– graph sparsifiers

algorithmic tools

efficiency considerations

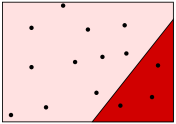
- data in the web and social-media are typically of extremely large scale (easily reach to billions)
- how to compute simple graph statistics?
- even quadratic algorithms are not feasible in practice

hashing and sketching

- probabilistic / approximate methods
- sketching : create sketches that summarize the data and allow to estimate simple statistics with small space
- hashing : hash objects in such a way that similar objects have larger probability of mapped to the same value than non-similar objects

estimator theorem

- consider a set of items U
- a fraction ρ of them have a specific property
- estimate ρ by sampling



- how many samples N are needed?

$$N \geq \frac{4}{\epsilon^2 \rho} \log \frac{2}{\delta}.$$

  for an ϵ-approximation with probability at least $1 - \delta$
- notice: it does not depend on $|U|$ (!)

homework

use the Chernoff bound to derive the estimator theorem

computing statistics on data streams

- $X = (x_1, x_2, \ldots, x_m)$ a sequence of elements
- each x_i is a member of the set $N = \{1, \ldots, n\}$
- $m_i = |\{j : x_j = i\}|$ the number of occurrences of i

define

$$F_k = \sum_{i=1}^{n} m_i^k$$

computing statistics on data streams

- $X = (x_1, x_2, \ldots, x_m)$ a sequence of elements
- each x_i is a member of the set $N = \{1, \ldots, n\}$
- $m_i = |\{j : x_j = i\}|$ the number of occurrences of i

  define
  $$F_k = \sum_{i=1}^{n} m_i^k$$
- F_0 is the number of distinct elements

computing statistics on data streams

- $X = (x_1, x_2, \ldots, x_m)$ a sequence of elements
- each x_i is a member of the set $N = \{1, \ldots, n\}$
- $m_i = |\{j : x_j = i\}|$ the number of occurrences of i

  define

$$F_k = \sum_{i=1}^{n} m_i^k$$

- F_0 is the number of distinct elements
- F_1 is the length of the sequence

computing statistics on data streams

- $X = (x_1, x_2, \ldots, x_m)$ a sequence of elements
- each x_i is a member of the set $N = \{1, \ldots, n\}$
- $m_i = |\{j \,:\, x_j = i\}|$ the number of occurrences of i

define

$$F_k = \sum_{i=1}^{n} m_i^k$$

- F_0 is the number of distinct elements
- F_1 is the length of the sequence
- F_2 index of homogeneity, size of self-join, and other applications

computing statistics on data streams

- How to compute the frequency moments using less than $O(n \log m)$ space?
- sketching: create a sketch that takes much less space and gives an estimation of F_k

[Alon et al., 1999]

estimating the number of distinct values (F_0)

[Flajolet and Martin, 1985]

- consider a bit vector of length $O(\log n)$
- upon seen x_i, set:
  - the 1st bit with probability $1/2$
  - the 2nd bit with probability $1/4$
  - ...
  - the i-th bit with probability $1/2^i$
- important: bits are set deterministically for each x_i
- let R be the index of the largest bit set
- return $Y = 2^R$

estimating number of distinct values (F_0)

Theorem. For every $c > 2$, the algorithm computes a number Y using $O(logn)$ memory bits, such that the probability that the ratio between Y and F_0 is not between $1/c$ and c is at most $2/c$

locality sensitive hashing

a family $\mathcal{H}$ is called (R, cR, p_1, p_2)-sensitive if for any two objects p and q

- if $d(p, q) \leq R$, then $\Pr_{\mathcal{H}}[h(p) = h(q)] \geq p_1$
- if $d(p, q) \geq cR$, then $\Pr_{\mathcal{H}}[h(p) = h(q)] \leq p_2$

interesting case when $p_1 > p_2$

locality sensitive hashing: example

- objects in a Hamming space $\{0, 1\}^d$ – binary vectors
- $\mathcal{H} : \{0, 1\}^d \to \{0, 1\}$ sample the *i* bit:
- $\mathcal{H} = \{h(x) = x_i \mid i = 1, \ldots, d\}$
- for two vectors x and y with distance r, it is

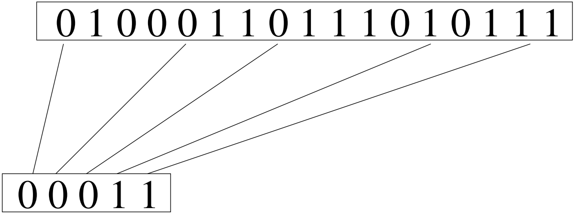
$$\Pr_{\mathcal{H}}[h(x) = h(y)] = 1 - \frac{r}{d}$$

- thus $p_1 = 1 - \frac{R}{d}$ and $p_2 = 1 - \frac{cR}{d}$
- gap between p_1 and p_2 too small
- probability amplification

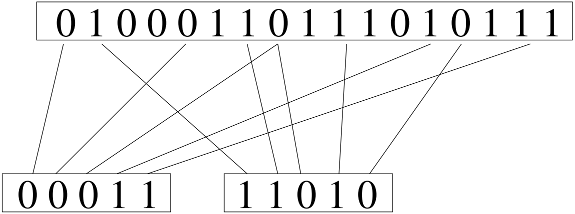
locality sensitive hashing: Hamming distance

0 1 0 0 0 1 1 0 1 1 1 0 1 0 1 1 1

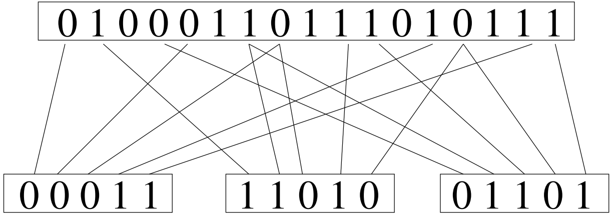
locality sensitive hashing: Hamming distance



locality sensitive hashing: Hamming distance



locality sensitive hashing: Hamming distance

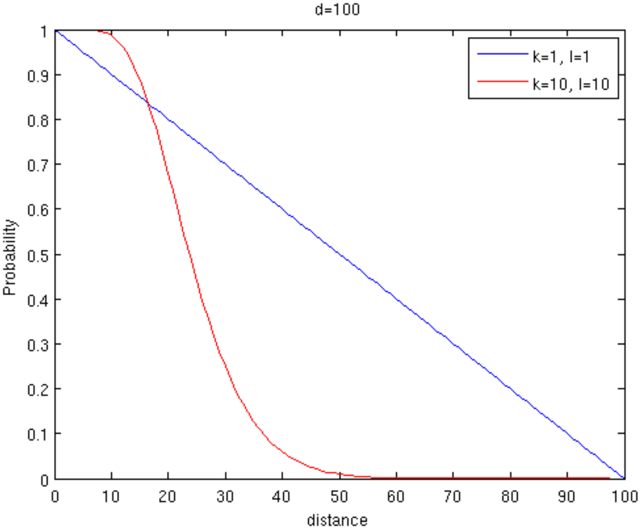


locality sensitive hashing: Hamming distance

Probability of collision

$$\Pr[h(x) = h(y)] = 1 - (1 - (1 - \frac{r}{d})^k)^l$$

locality sensitive hashing: Hamming distance



homework

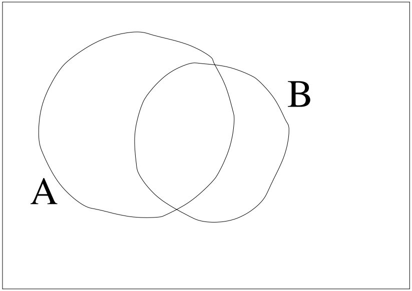
how to apply the locality sensitive hashing for vectors of integers, not just binary vectors?

vectors $\mathbf{x} = \{x_1, \ldots, x_d\}$

L_1 distance $||\mathbf{x} - \mathbf{y}||_1 = \sum_{i=1}^{d} |x_i - y_i|$

Jaccard coefficient

- for two sets $A, B \subseteq U$ define $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$
- measure of similarity of the sets



- can we design a locality sensitive hashing family for Jaccard?

min-wise independent permutations

- $\pi : U \to U$ a random permutation of U
- $h(A) = \min\{\pi(x) \mid x \in A\}$

min-wise independent permutations

- $\pi : U \to U$ a random permutation of U
- $h(A) = \min\{\pi(x) \mid x \in A\}$
- then

$$\Pr[h(A) = h(B)] = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- amplify the probability as before:
  - repeat many times,
  - concatenate into blocks
  - consider objects similar if they collide in at least one block

homework

show that for $h(A) = \min\{\pi(x) \mid x \in A\}$, with π a random permutation, it is

$$\Pr[h(A) = h(B)] = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

homework

design a locality-sensitive hashing scheme for vectors according to the cosine similarity measure

vectors $\mathbf{x} = \{x_1, \ldots, x_d\}$

distance $1 - \cos(\mathbf{x}, \mathbf{y}) = 1 - \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||_2 \, ||\mathbf{y}||_2}$

applications of the algorithmic tools to real scenarios

clustering coefficient and triangles

clustering coefficient

$$C = \frac{3 \times \text{number of triangles in the network}}{\text{number of connected triples of vertices}}$$

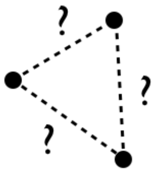
- how to compute it?
- how to compute the number of triangles in a graph?
- assume that the graph is very large, stored in disk

[Buriol et al., 2006]

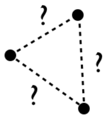
- count triangles when graph is seen as a data stream
- two models:
  - edges are stored in any order
  - edges in order : all edges incident to one vertex are stored sequentially

counting triangles

- brute-force algorithm is checking every triple of vertices
- obtain an approximation by sampling triples



sampling algorithm for counting triangles



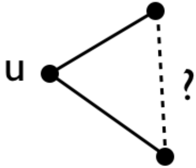
- how many samples are required?
- let T be the set of all triples and
  T_i the set of triples that have i edges, $i = 0, 1, 2, 3$
- by the estimator theorem, to get an ϵ-approximation, with probability $1 - \delta$, the number of samples should be

$$N \geq O(\frac{|T|}{|T_3|}\frac{1}{\epsilon^2}\log\frac{1}{\delta})$$

- but $|T|$ can be very large compared to $|T_3|$

counting triangles

- incidence model : all edges incident to each vertex appear in order in the stream
- sample connected triples



sampling algorithm for counting triangles

- incidence model
- consider sample space $\mathcal{S} = \{b\text{-}a\text{-}c \mid (a,b),(a,c) \in E\}$
- $|\mathcal{S}| = \sum_i d_i(d_i - 1)/2$

1: sample $X \subseteq \mathcal{S}$ (paths b-a-c)
2: estimate fraction of X for which edge (b,c) is present
3: scale by $|\mathcal{S}|$

- gives (ϵ, δ) approximation

counting triangles — incidence stream model

SAMPLETRIANGLE [Buriol et al., 2006]

1st pass

count the number of paths of length 2 in the stream

2nd pass

uniformly choose one path (a, b, c)

3rd pass

if $((b, c) \in E)$ $\beta = 1$ else $\beta = 0$

return β

counting triangles — incidence stream model

SampleTriangle [Buriol et al., 2006]

1st pass

count the number of paths of length 2 in the stream

2nd pass

uniformly choose one path (a, b, c)

3rd pass

if $((b, c) \in E)$ $\beta = 1$ else $\beta = 0$

return β

we have $\mathrm{E}[\beta] = \frac{3|T_3|}{|T_2|+3|T_3|}$, with $|T_2| + 3|T_3| = \sum_u \frac{d_u(d_u-1)}{2}$, so

$$|T_3| = \mathrm{E}[\beta] \sum_u \frac{d_u(d_u-1)}{6}$$

and space needed is $O((1 + \frac{|T_2|}{|T_3|}) \frac{1}{\epsilon^2} \log \frac{1}{\delta})$

properties of the sampling space

it should be possible to

- estimate the size of the sampling space
- sample an element uniformly at random

homework

1. compute triangles in 3 passes when edges appear in arbitrary order
2. compute triangles in 1 pass when edges appear in arbitrary order
3. compute triangles in 1 pass in the incidence model

triangle sparsifiers

[Tsourakakis et al., 2011]

- start with graph $G(V, E)$
- use sparsification parameter p
- pick a random subset E' of edges
  each edge is selected with probability p
- T_3' = # triangles on graph $G'(V, E')$
- return $T_3 = T_3'/p^3$

triangle sparsifiers

[Tsourakakis et al., 2011]

- start with graph $G(V, E)$
- use sparsification parameter p
- pick a random subset E' of edges
  each edge is selected with probability p
- T_3' = # triangles on graph $G'(V, E')$
- return $T_3 = T_3'/p^3$

- T_3 is highly concentrated around the true number
  of triangles

counting graph minors

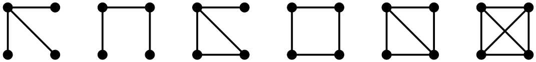
counting other minors

- count all minors in a very large graphs
  - connected subgraphs
  - size 3 and 4
  - directed or undirected graphs

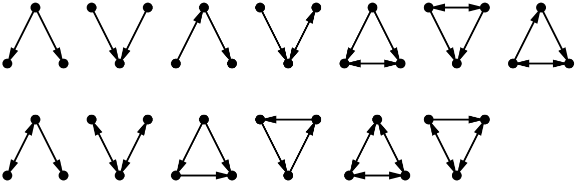
- why?
- modeling networks, “signature” structures
  e.g., copying model
- anomaly detection, e.g., spam link farms
  [Alon, 2007, Bordino et al., 2008]

counting minors in large graphs

- characterize a graph by the distribution of its minors



all undirected minors of size 4



all directed minors of size 3

sampling algorithm for counting triangles

- incidence model
- consider sample space $\mathcal{S} = \{b\text{-}a\text{-}c \mid (a, b), (a, c) \in E\}$
- $|\mathcal{S}| = \sum_i d_i(d_i - 1)/2$

1: sample $X \subseteq \mathcal{S}$ (paths $b\text{-}a\text{-}c$)
2: estimate fraction of X for which edge (b, c) is present
3: scale by $|\mathcal{S}|$

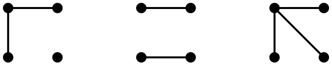
- gives (ϵ, δ) approximation

adapting the algorithm

sampling spaces:

- 3-node directed

- 4-node undirected



are the sampling space properties satisfied?

datasets

graph class	type	# instances
synthetic	un/directed	39
wikipedia	un/directed	7
webgraphs	un/directed	5
cellular	directed	43
citation	directed	3
food webs	directed	6
word adjacency	directed	4
author collaboration	undirected	5
autonomous systems	undirected	12
protein interaction	undirected	3
US road	undirected	12

clustering of undirected graphs

assigned to	0	1	2	3	4	5	6
AS graph	12	0	0	0	0	0	0
collaboration	0	0	3	2	0	0	0
protein	1	0	0	1	0	0	1
road-graph	0	12	0	0	0	0	0
wikipedia	0	0	0	0	2	5	0
synthetic	11	0	0	0	0	0	28
webgraph	2	0	0	1	0	0	0

clustering of directed graphs

feature class	accuracy compared to ground truth
standard topological properties (81)	0.74%
minors of size 3	0.78%
minors of size 4	0.84%
minors of size 3 and 4	0.91%

local statistics

compute local statistics in large graphs

- our goal: compute triangle counts for all vertices
- local clustering coefficient and related statistics
- motivation
  - motifs can be used to characterize network families [Alon, 2007, Bordino et al., 2008]
  - analysis of social or biological networks
  - thematic relationships in the web
  - web spam
- applications: spam detection and content quality analysis in social media

semi-streaming model

[Feigenbaum et al., 2004]

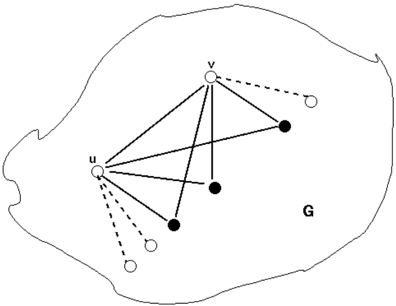
- data stream model (constant memory) too restrictive
- graph stored in secondary memory as adjacency or edge list
- ✗ no random access possible
- $O(N \log N)$ bits available in main memory
  - limited amount of information per vertex
  - ✗ not enough to store edges in main memory
- limited (constant or $O(\log N)$) number of passes
- compute counts for all vertices concurrently

two algorithms

1. external memory
   - keep a counter for each vertex (main memory)
   - keep a counter for each edge (secondary memory)

2. main memory
   - keep a counter for each vertex

number of triangles for edges and nodes



- neighbors: $N(u) = \{v : (u, v) \in E\}$
- degree: $d(u) = |N(u)|$
- edge triangles: $T_{uv} = |N(u) \cap N(v)|$
- vertex triangles: $T(u) = \frac{1}{2} \sum_{v \in N(u)} T_{uv}$

computing triangles : idea

- consider the Jaccard coefficient between two sets A and B:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- if we knew $J(N(u), N(v)) = J$, then:

$$T_{uv} = |N(u) \cap N(v)| = \frac{J}{J+1}(|N(u)| + |N(v)|)$$

- and then:

$$T(u) = \frac{1}{2} \sum_{v \in N(u)} T_{uv}$$

computing triangles : idea

we want:

$$T_{uv} = |N(u) \cap N(v)| = \frac{J}{J+1}(|N(u)| + |N(v)|)$$

approximate the Jaccard coefficient:

- m independent trials
- Z_{uv} : # times that $\min \pi(N(u)) = \min \pi(N(v))$

use the estimator:

$$\overline{T}_{uv} = \frac{Z_{uv}}{Z_{uv} + m}(|N(u)| + |N(v)|)$$

external-memory algorithm

- semi-stream model
- keep vertex min-hash values (in memory)
- keep edge counters (on disk)
- use edge counters to estimate number of triangles (and local clustering coefficient)

external-memory algorithm

Z = **0**
for i: 1 . . . m **do** {independent trials}
    for u : 1 . . . $|V|$ **do** {assign labels}
        $l_i(u) = \text{hash}_i(u)$ {Min-wise linear permutation}
    end for

external-memory algorithm

Z = **0**
for i: 1 ... m **do** {independent trials}
    for u : 1 ... $|V|$ **do** {assign labels}
        $l_i(u)$ = hash$_i(u)$ {Min-wise linear permutation}
    end for
    for u : 1 ... $|V|$ **do** {compute fingerprints}
        $F_i(u) = \min_{v \in N(u)} l_i(u)$
    end for{1 scan of G}

external-memory algorithm

```
Z = 0
for i: 1 ... m do {independent trials}
    for u : 1 ... |V| do {assign labels}
        l_i(u) = hash_i(u) {Min-wise linear permutation}
    end for
    for u : 1 ... |V| do {compute fingerprints}
        F_i(u) = min_{v∈N(u)} l_i(u)
    end for{1 scan of G}
    for u : 1 ... |V| do {update counters}
        for v ∈ N(u) do
            if (F_i(u) = F_i(v)) then {minima are equal}
                Z_uv = Z_uv + 1 {Z_uv's stored on disk}
            end if
        end for
    end for
end for
```

implementation

- $\text{hash}_i(x)$ is, e.g., a linear hash function ($a_i x + b_i \mod p$)
- for every i, the $F_i(u)$'s can be kept in main memory
- the Z_{uv}'s must be stored on disk
  - for every i, updating Z_{uv} requires access to disk
  - computing counters most expensive operation

main-memory algorithm

- replace:

$$\overline{T}_{uv} = \frac{Z_{uv}}{Z_{uv} + m}(|N(u)| + |N(v)|)$$

- by the estimator for $|N(u) \cap N(v)|$:

$$\tilde{T}_{uv} = \frac{Z_{uv}}{\frac{2}{3}m}(N(u) + N(v))$$

- and estimator for $T(u)$:

$$\tilde{T}(u) = \frac{1}{3m} \sum_{v \in N(u)} Z_{uv}(N(u) + N(v)) = \frac{1}{3m} Z_u$$

- Z_u sums $d(u) + d(v)$ if $\min \pi(N(u)) = \min \pi(N(v))$
- only one counter per node

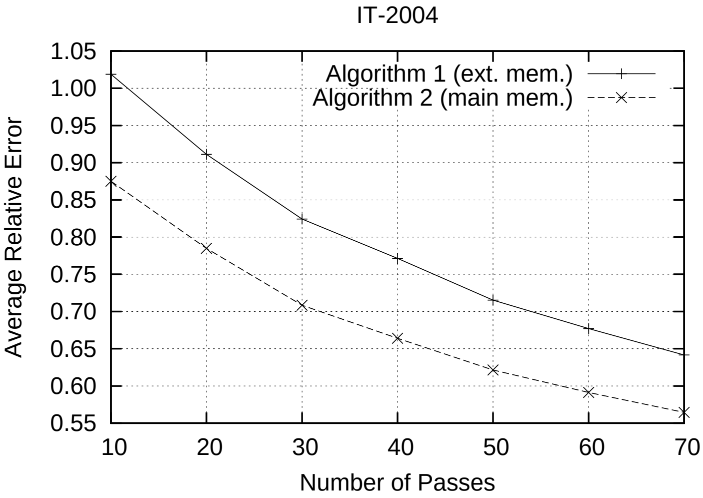
main-memory algorithm

Z = 0
for i: 1 … m **do** {Independent trials}
  for u : 1 … $|V|$ **do** {Assign labels}
    $l_i(u) = \text{hash}_i(u)$
  end for
  for u : 1 … $|V|$ **do** {Compute fingerprints}
    $F_i(u) = \min_{v \in V(u)} l_i(u)$
  end for{1 scan of G}
  for u : 1 … $|V|$ **do** {Update counters}
    for $v \in N(u)$ **do**
      if $F_i(u) == F_i(v)$ **then** {Minima are equal}
        $Z_u = Z_u + d(u) + d(v)$ {Z_u's in **main memory**}
      end if
    end for
  end for
end for

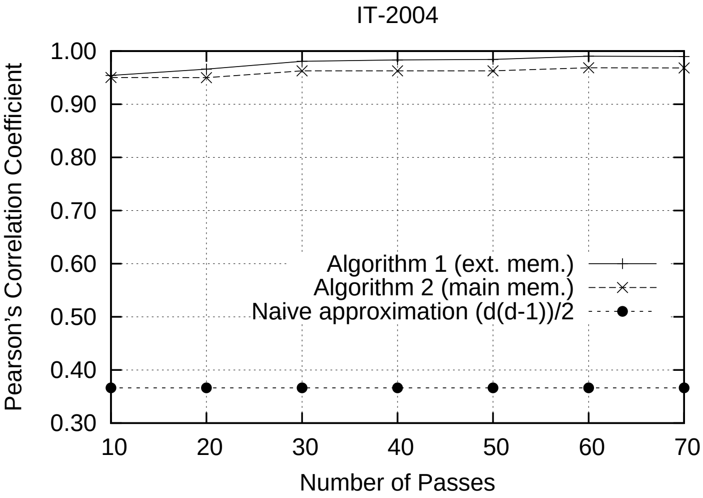
experimental results

Graph	Nodes	Edges	Algorithm 1 (ext. mem.)	Algorithm 2 (main mem.)
WB-2001	118M	1.7G	10 hr 20 min	3 hr 40 min
IT-2004	41M	2.1G	8 hr 20 min	5 hr 30 min
UK-2006	77M	5.3G	20 hr 30 min	13 hr 10 min

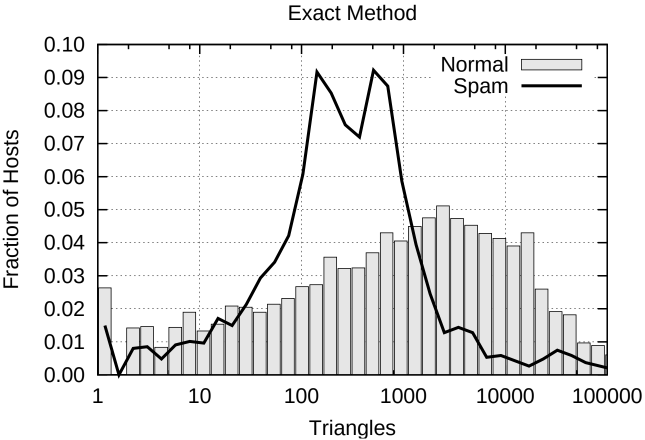
quality of approximation



quality of approximation

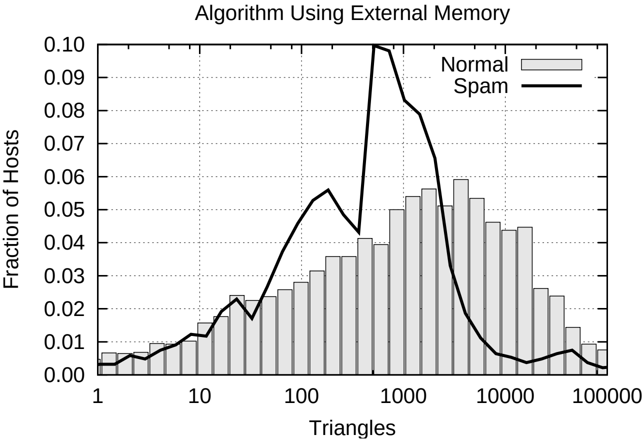


applications : spam detection



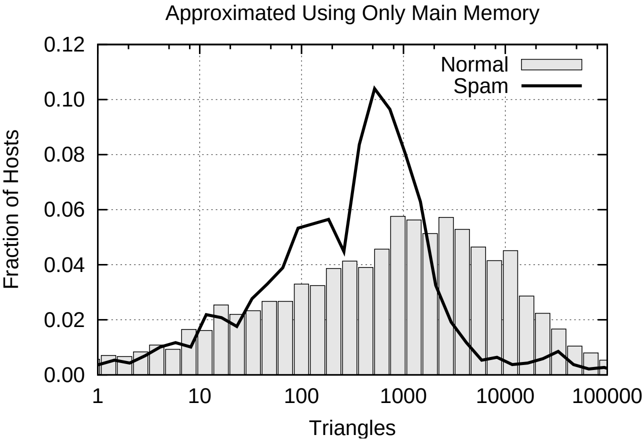
Separation of non-spam and spam hosts in the histogram of triangles

applications : spam detection



Separation of non-spam and spam hosts in the histogram of triangles

applications : spam detection



Separation of non-spam and spam hosts in the histogram of triangles

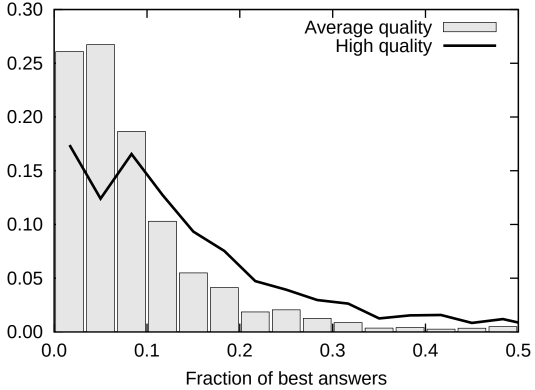
applications : spam detection

number of triangles feature is ranked 60-th out of 221 for spam detection

applications : content quality in yahoo! answers

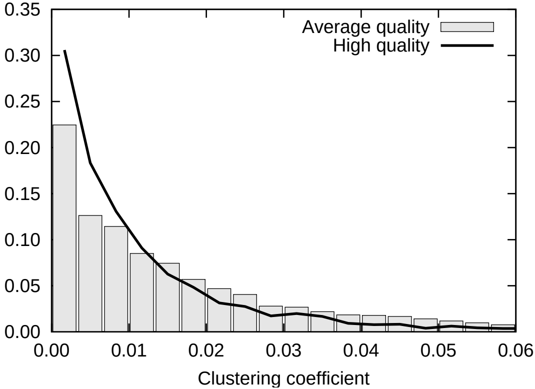
- Yahoo! answers, a question-answering portal
- consider the graph with edges (u, v) if user u has answered a question of user v
- consider "high quality" users those who have given a best answer to a random sample of questions
- predict high-quality users based on their local structure

applications : content quality in yahoo! answers



Separation of users who have provided questions/answers of high quality with users who have provided questions/answers of normal quality in terms of fraction of best answers

applications : content quality in yahoo! answers



Separation of users who have provided questions/answers of high quality with users who have provided questions/answers of normal quality in terms of local clustering coefficient

graph distance distributions

small-world phenomena

small worlds : graphs with short paths

- Stanley Milgram (1933-1984)
  "The man who shocked the world"
- obedience to authority (1963)
- small-world experiment (1967)

Milgram's experiment

- 300 people (starting population) are asked to dispatch a parcel to a single individual (target)
- the target was a Boston stockbroker
- the starting population is selected as follows:
  - 100 were random Boston inhabitants (group A)
  - 100 were random Nebraska strockbrokers (group B)
  - 100 were random Nebraska inhabitants (group C)

Milgram's experiment

- rules of the game :
- parcels could be directly sent only to someone the sender knows personally
- 453 intermediaries happened to be involved in the experiments (besides the starting population and the target)

Milgram's experiment

questions Milgram wanted to answer:

1. how many parcels will reach the target?

2. what is the distribution of the number of hops required to reach the target?

3. is this distribution different for the three starting subpopulations?

Milgram's experiment

answers to the questions

1. how many parcels will reach the target?
   29%
2. what is the distribution of the number of hops required to reach the target?
   average was 5.2
3. is this distribution different for the three starting subpopulations?
   YES: average for groups A/B/C was 4.6/5.4/5.7

chain lengths

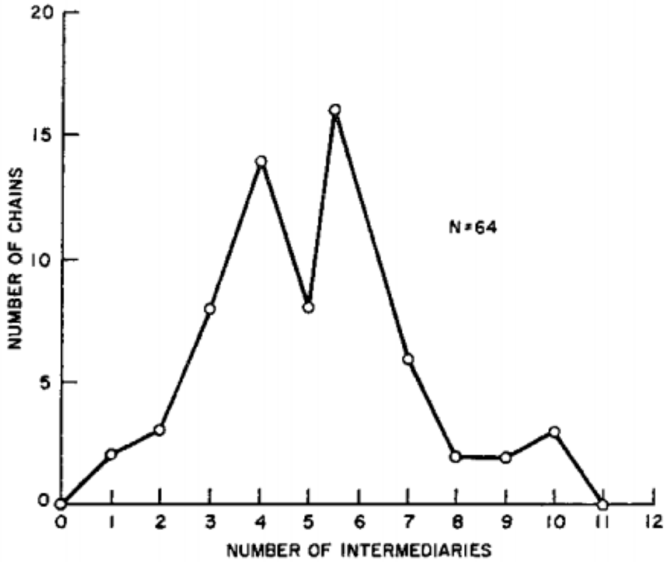


FIGURE 1

measuring what?

but what did Milgram's experiment reveal, after all?

1. the the world is small
2. that people are able to exploit this smallness

graph distance distribution

- obtain information about a large graph, i.e., social network
- macroscopic level
- distance distribution
  - mean distance
  - median distance
  - diameter
  - effective diameter
  - ...

graph distance distribution

- given a graph, $d(x, y)$ is the length of the shortest path from x to y, defined as ∞ if one cannot go from x to y
- for undirected graphs, $d(x, y) = d(y, x)$
- for every t, count the number of pairs (x, y) such that $d(x, y) = t$
- the fraction of pairs at distance t is a distribution

exact computation

how can one compute the distance distribution?

- weighted graphs: Dijkstra (single-source: $O(m \log n)$),
- Floyd-Warshall (all-pairs: $O(n^3)$)
- in the unweighted case:
  - a single BFS solves the single-source version of the problem: $O(m)$
  - if we repeat it from every source: $O(nm)$

sampling pairs

- sample at random pairs of nodes (x, y)
- compute $d(x, y)$ with a BFS from x
- (possibly: reject the pair if $d(x, y)$ is infinite)

sampling pairs

- for every t, the fraction of sampled pairs that were found at distance t are an estimator of the value of the probability mass function
- takes a BFS for every pair — $O(m)$

sampling sources

- sample at random a source t
- compute a full BFS from t

sampling sources

- it is an unbiased estimator only for undirected and connected graphs
- uses anyway BFS...
  - ...not cache friendly
  - ... not compression friendly

idea : diffusion

[Palmer et al., 2002]

- let $B_t(x)$ be the ball of radius t around x (the set of nodes at distance $\leq t$ from x)
- clearly $B_0(x) = \{x\}$
- moreover $B_{t+1}(x) = \bigcup_{(x,y)} B_t(y) \bigcup \{x\}$
- so computing B_{t+1} from B_t just takes a single (sequential) scan of the graph

easy but costly

- every set requires $O(n)$ bits, hence $O(n^2)$ bits overall
- easy but costly
- too many!
- what about using approximated sets?
- we need probabilistic counters, with just two primitives: add and size
- very small!

estimating the number of distinct values (F_0)

- [Flajolet and Martin, 1985]
- consider a bit vector of length $O(\log n)$
- upon seen x_i, set:
  - the 1st bit with probability $1/2$
  - the 2nd bit with probability $1/4$
  - ...
  - the i-th bit with probability $1/2^i$
- important: bits are set deterministically for each x_i
- let R be the index of the largest bit set
- return $Y = 2^R$

ANF

- probabilistic counter for approximating the number of distinct values [Flajolet and Martin, 1985]
- ANF algorithm [Palmer et al., 2002]
  uses the original probabilist counters
- HyperANF algorithm [Boldi et al., 2011]
  uses HyperLogLog counters [Flajolet et al., 2007]

HyperANF

- HyperLogLog counter [Flajolet et al., 2007]
- with 40 bits you can count up to 4 billion with a standard deviation of 6%
- remember: one set per node

implementation tricks

[Boldi et al., 2011]

- use broad-word programming to compute union efficiently
- systolic computation for on-demand updates of counters
- exploit micro-parallelization of multicore architectures

performance

- HADI, a Hadoop-conscious implementation of ANF [Kang et al., 2011]
- takes 30 minutes on a 200K-node graph (on one of the 50 world largest supercomputers)
- HyperANF does the same in 2.25min on a workstation (20 min on a laptop).

experiments on facebook

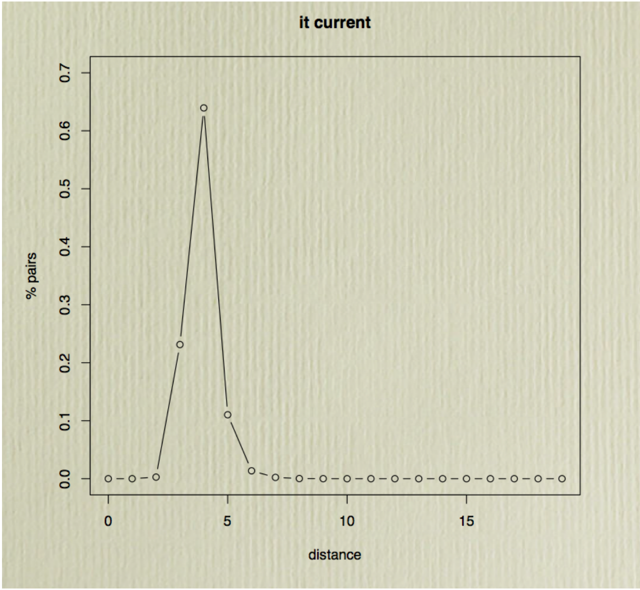
[Backstrom et al., 2011]

considered only active users

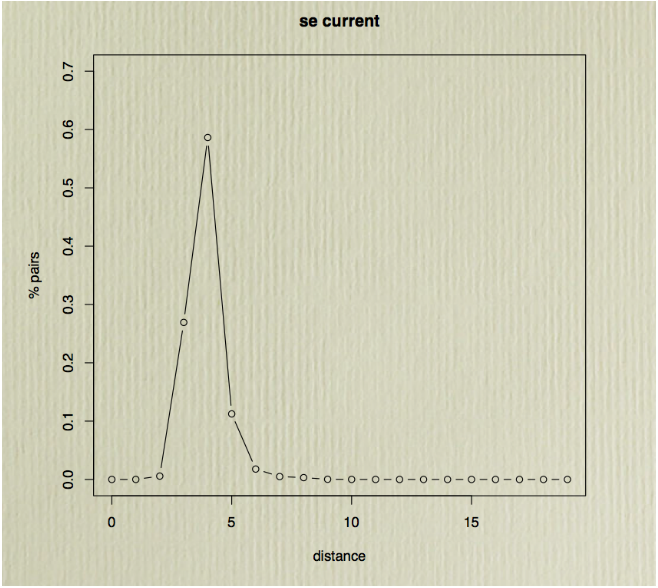
- it : only italian users
- se : only swedish users
- it + se : only italian and swedish users
- us : only US users
- the whole facebook (750m nodes)

based on users current geo-IP location

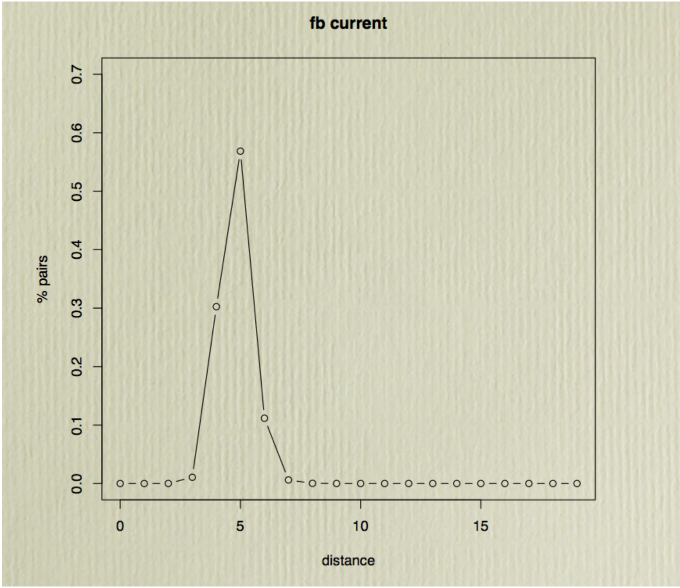
distance distribution (it)



distance distribution (se)



distance distribution (fb)



average distance

	2008	2012
it	6.58	3.90
se	4.33	3.89
it+se	4.90	4.16
us	4.74	4.32
fb	5.28	4.74

fb 2012 : 92% pairs are reachable!

effective diameter

	2008	2012
it	9.0	5.2
se	5.9	5.3
it+se	6.8	5.8
us	6.5	5.8
fb	7.0	6.2

actual diameter

	2008	2012
it	> 29	$= 25$
se	> 16	$= 25$
it+se	> 21	$= 27$
us	> 17	$= 30$
fb	> 17	> 58

breaking the news

HOME PAGE | TODAY'S PAPER | VIDEO | MOST POPULAR | TIMES TOPICS

Log In | Register Now | Help

The New York Times

Business Day

Technology

Search All NYTimes.com

Go

WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH | SPORTS | OPINION | ARTS | STYLE | TRAVEL | JOBS | REAL ESTATE | AUTOS

Separating You and Me? 4.74 Degrees

By JOHN MARKOFF and SOMINI SENGUPTA

Published: November 21, 2011

The world is even smaller than you thought.

Enlarge This Image

Cornell News Service

Jon Kleinberg of Cornell said weak ties could be important.

Adding a new chapter to the research that cemented the phrase "six degrees of separation" into the language, scientists at Facebook and the University of Milan reported on Monday that the average number of acquaintances separating any two people in the world was not six but 4.74.

The original "six degrees" finding, published in 1967 by the psychologist Stanley Milgram, was drawn from 296 volunteers who were asked to send a message by postcard, through friends and then friends of friends, to a specific person in a Boston suburb.

RECOMMEND

TWITTER

LINKEDIN

SIGN IN TO E-MAIL

PRINT

REPRINTS

SHARE

Log in to see what your friends are sharing on nytimes.com. Privacy Policy | What's This?

Log In With Facebook

What's Popular Now

Marvin Hamlisch, Composer, Dies at 68

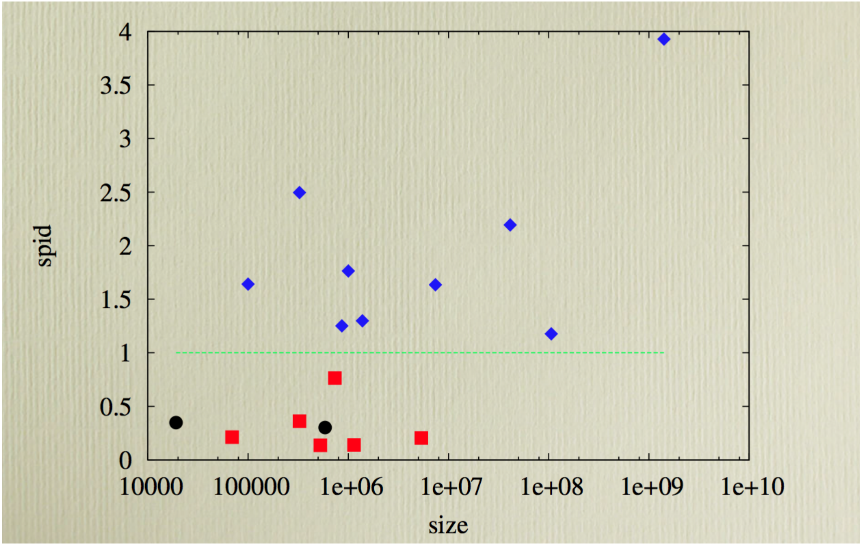
France's 'les Riches' Vow to Leave if 75% Tax Rate Is Passed

another application : spid

[Boldi et al., 2011]

- spid : shortest-paths index of dispersion
- the ratio between variance and average in the distance distribution
- spid < 1 : the distribution is subdispersed
- spid > 1 : is superdispersed
- web graphs and social networks have different spid!

spid plot


spid
size
4
3.5
3
2.5
2
1.5
1
0.5
0
10000
100000
1e+06
1e+07
1e+08
1e+09
1e+10

the spid conjecture

- [Boldi et al., 2011] conjecture that spid is able to tell social networks from web graphs
- average distance alone would not suffice: it is very changeable and depends on the scale
- spid, instead, seems to have a clear cutpoint at 1
- what is facebook spid?

the spid conjecture

- [Boldi et al., 2011] conjecture that spid is able to tell social networks from web graphs
- average distance alone would not suffice: it is very changeable and depends on the scale
- spid, instead, seems to have a clear cutpoint at 1
- what is facebook spid? 0.093

indexing distances in large graphs

shortest-path distances in large graphs

- input: consider a graph $G = (V, E)$
- and nodes s and t in V
- goal: compute the shortest-path distance $d(s, t)$ from s to t
- do it very fast

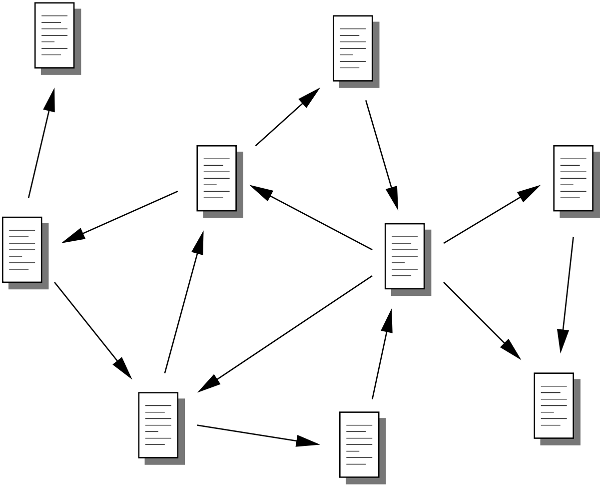
well-studied problem

different strategies

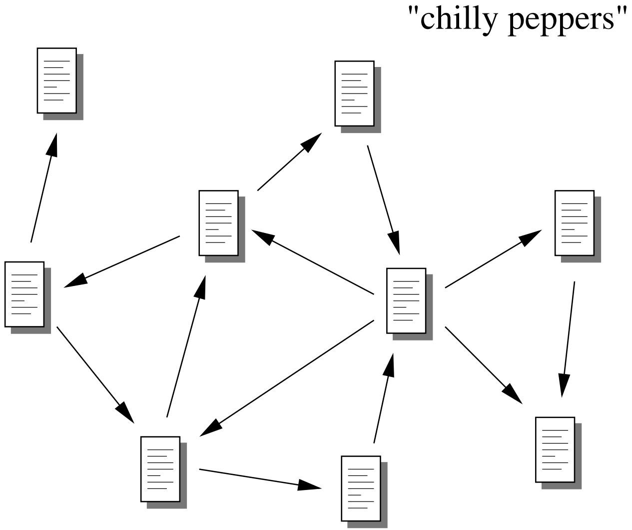
- lazy
  - compute shortest path at query time
  - Dijkstra, BFS
  - no precomputation
  - BFS takes $O(m)$
  - too expensive for large graphs
- eager
  - precompute all-pairs shortest paths
  - Floyd-Warshall, matrix multiplication
  - $O(n^3)$ precomputation, $O(n^2)$ storage
  - too large to store

applications of shortest-path queries

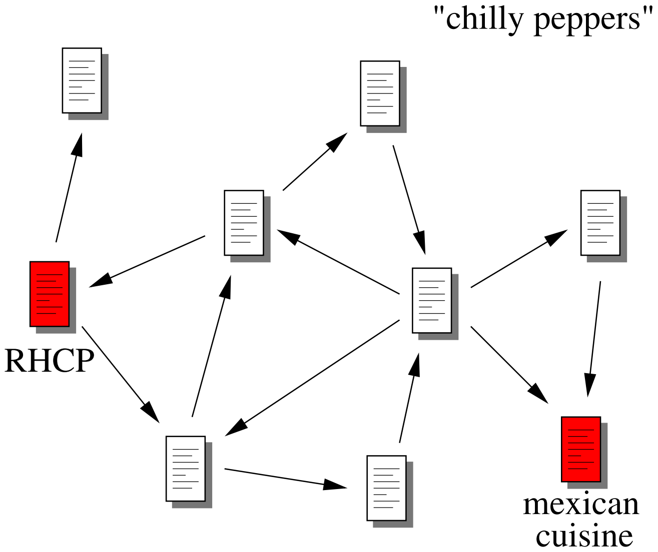
searching in graphs — I. context-sensitive search



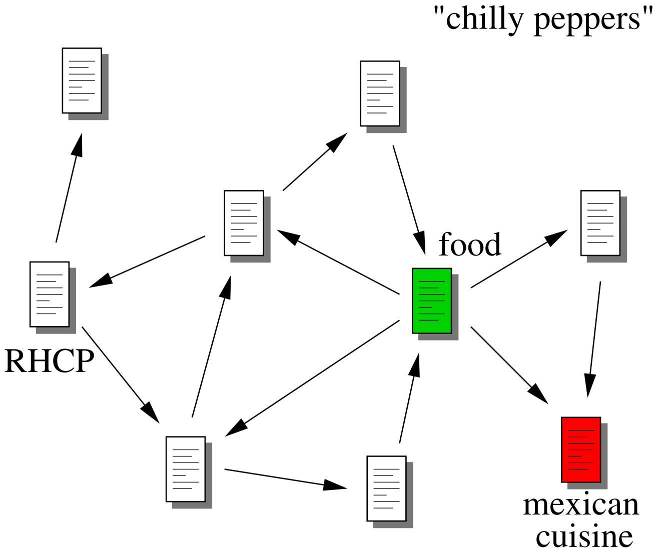
searching in graphs — I. context-sensitive search



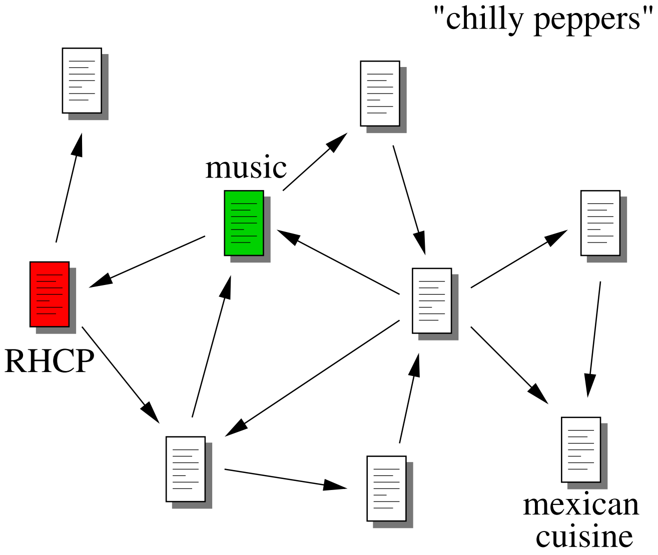
searching in graphs — I. context-sensitive search



searching in graphs — I. context-sensitive search



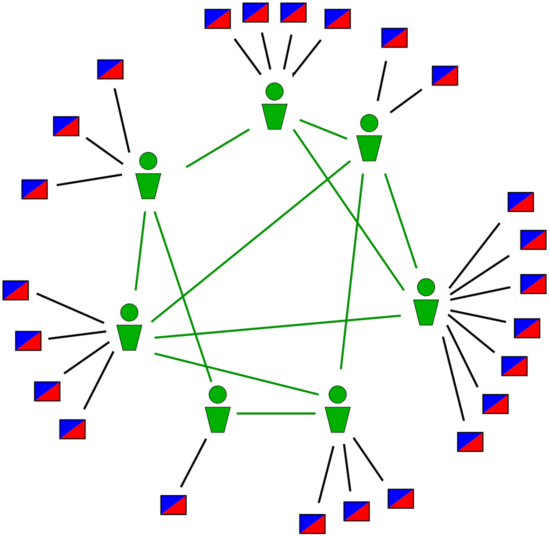
searching in graphs — I. context-sensitive search



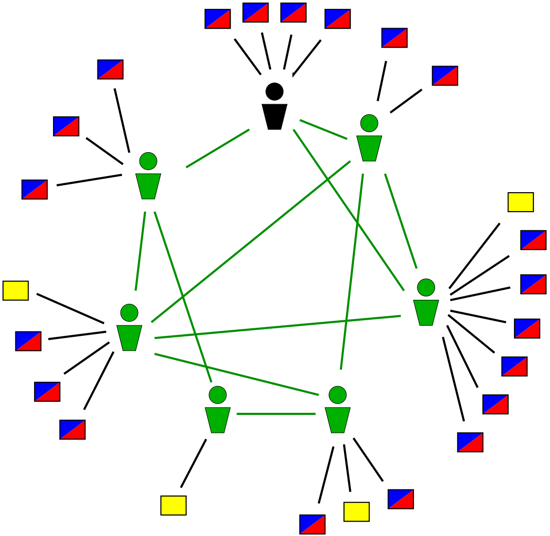
searching in graphs — I. context-sensitive search

- customize search results to the user's current page or recent history of pages have visited
- increasing relevance of answers
- disambiguation
- suggesting links to wikipedia editors

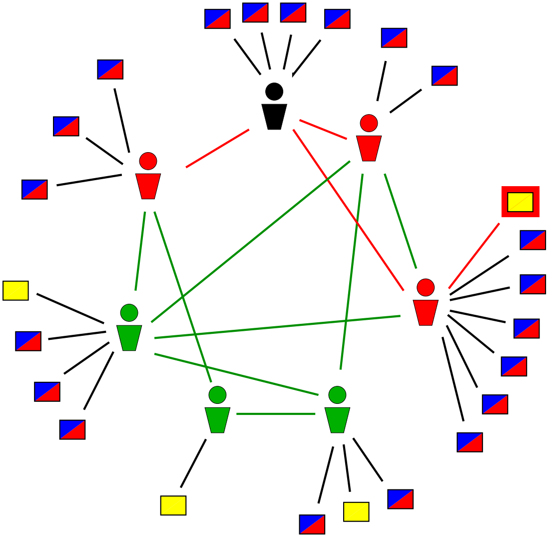
searching in graphs — II. social search



searching in graphs — II. social search



searching in graphs — II. social search



searching in graphs — II. social search

- consider more information than just contacts
  - preferences
  - geographical information
  - comments
  - favorites
  - tags
  - etc.

machine-learning approach

- learn a ranking function that combines a large number of features

  content-based features:

  - TF/IDF, BM25, etc., as in traditional IR and web search
  - content similarity between the querying node and a target node

  link-based features:

  - PageRank
  - shortest-path distance from the querying node to a target node
  - spectral distance from the querying node to a target node
  - graph-based similarity measures
  - context-specific PageRank

well-studied problem

different strategies

- lazy
  - compute shortest path at query time
  - Dijkstra, BFS
  - no precomputation
  - BFS takes $O(m)$
  - too expensive for large graphs
- eager
  - precompute all-pairs shortest paths
  - Floyd-Warshall, matrix multiplication
  - $O(n^3)$ precomputation, $O(n^2)$ storage
  - too large to store

anything in between?

- is there a smooth tradeoff between

  $\langle O(1), O(m) \rangle$ and $\langle O(n^2), O(1) \rangle$

distance oracles

[Thorup and Zwick, 2005]

- given a graph $G = (V, E)$
- an (α, β)-approximate distance oracle is a data structure S that
- for a query pair of nodes (u, v), S returns $d_S(u, v)$ s.t.

$$d(u, v) \leq d_S(u, v) \leq \alpha\, d(u, v) + \beta$$

- α called stretch or distortion
- consider the preprocessing time, the required space, and the query time

distance oracles

[Thorup and Zwick, 2005]

- given k, construct an oracle with storage $O(kn^{1+1/k})$, query time $O(k)$, stretch $2k-1$
- $k = 1$
  $\Rightarrow$ APSP
- $k = \log n$
  $\Rightarrow$ storage $O(n \log n)$, query time $O(\log n)$, stretch $O(\log n)$

distance oracles — preprocessing

[Das Sarma et al., 2010]

1. $r = \lfloor \log |V| \rfloor$
2. sample $r + 1$ sets of sizes $1, 2, 2^2, 2^3, \ldots, 2^r$
3. call the sampled sets $S_0, S_1, \ldots, S_r$
4. for each node u and each set S_i compute (w_i, δ_i),

   where $\delta_i = d(u, w_i) = \min_{v \in S_i}\{d(u, v)\}$
5. $\textsc{Sketch}[u] = \{(w_0, \delta_0), \ldots, (w_r, \delta_r)\}$
6. repeat k times

distance oracles — query processing

[Das Sarma et al., 2010]

given query (u, v)

1. obtain $\text{SKETCH}[u]$ and $\text{SKETCH}[v]$
2. find the set of common nodes w in $\text{SKETCH}[u]$ and $\text{SKETCH}[v]$
3. for each common node w, compute $d(u, w)$ and $d(w, v)$
4. return the minimum of $d(u, w) + d(w, v)$, taken over all common node w's
5. if no common w is present, then return ∞

landmark-based approach

- precompute: distance from each node to a fixed landmark l
- then

$$|d(s,l) - d(t,l)| \leq d(s,t) \leq d(s,l) + d(l,t)$$

- precompute: distances to d landmarks, $l_1, \ldots, l_d$

$$\max_i |d(s,l_i) - d(t,l_i)| \leq d(s,t) \leq \min_i (d(s,l_i) + d(l_i,t))$$

- obtain a range estimate in time $O(d)$ (i.e., constant)

landmark-based approach

- motivated by indexing general metric spaces
- used for estimating latency in the internet [Ng and Zhang, 2008]
- typically randomly chosen landmarks

theoretical results

[Kleinberg et al., 2004]

- random landmarks can provide distance estimates with distortion $(1 + \delta)$ for a fraction of at least $(1 - \epsilon)$ of pairs
- number of landmarks required depends on ϵ, δ, and the doubling dimension k of the metric space

approximation guarantee in practice

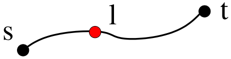
what does a logarithmic approximation guarantee mean in a small-world graph?

the landmark selection problem

how to choose good landmarks in practice?

good landmarks

if



then $d(s,t) = d(s,l) + d(l,t)$

if

then $|d(s,l) - d(t,l)| = d(s,t)$

good (upper-bound) landmarks

- a landmark l covers a pair (s, t) if l is on a shortest path from s to t
- problem definition: find a set $L \subseteq V$ of k landmarks that cover as many pairs $(s, t) \in V \times V$ as possible

- NP-hard
- for $k = 1$: the node with the highest centrality betweenness
- for $k > 1$: apply a "natural" set-cover approach (but $O(n^3)$)

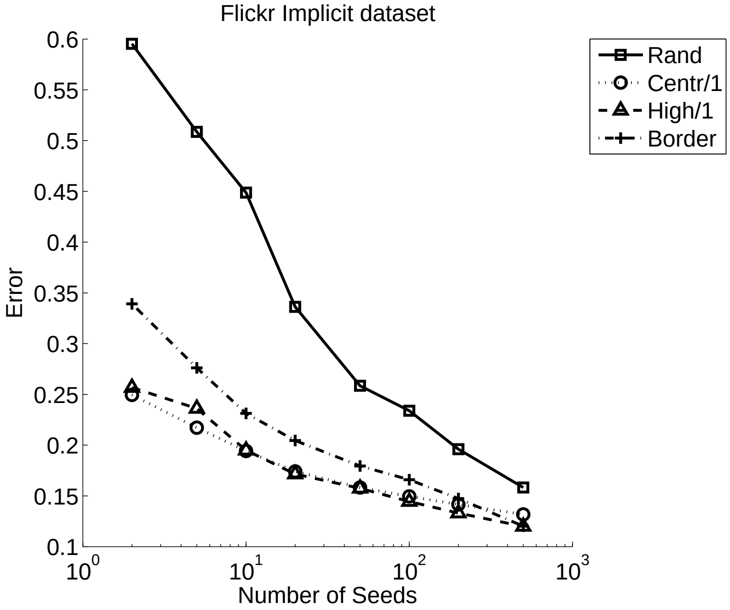
landmark selection heuristics

- high-degree nodes
- high-centrality nodes
- “constrained” versions
  - once a node is selected none of its neighbors is selected
- “clustered” versions
  - cluster the graph and select one landmark per cluster
  - select landmarks on the “borders” between clusters

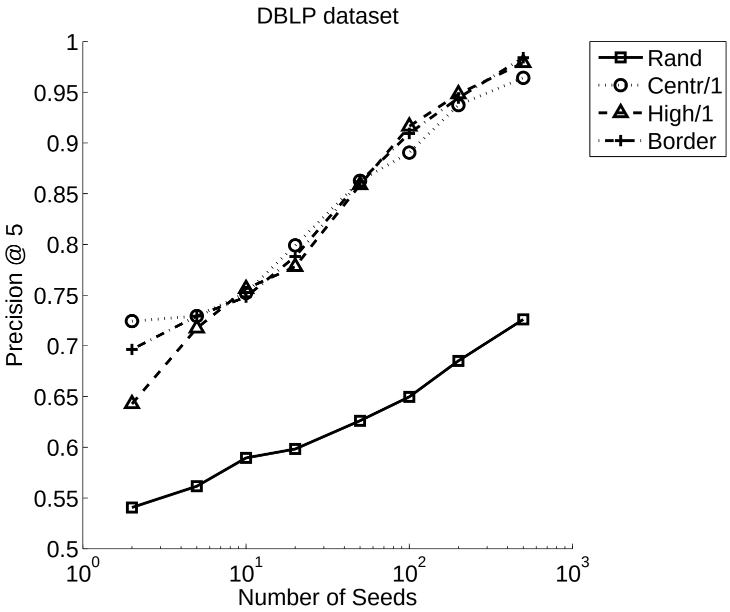
datasets

	# nodes	# edges	median distance	effective diameter	clustering coefficient
flickr	801 K	8 M	5	8	0.11
DBLP	226 K	716 K	9	13	0.47

flickr-implicit — distance error

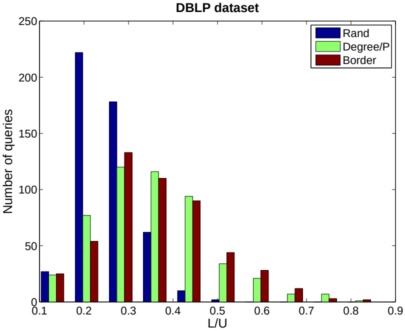


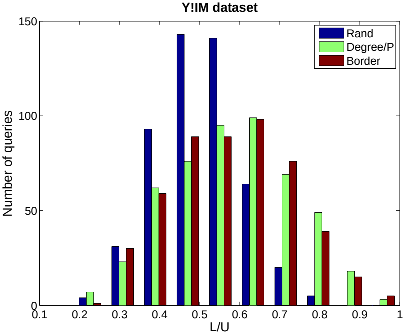
DBLP — **precision @ 5**



triangulation task

[Kleinberg et al., 2004]





comparing with exact algorithm

[Goldberg and Harrelson, 2005]

landmarks (10%)	Fl.-E	Fl.-I	Wiki	DBLP	Y!IM
Method	CENT	CENT	CENT/P	BORD/P	BORD/P
Landmarks used	20	100	500	50	50
Nodes visited	1	1	1	1	1
Operations	20	100	500	50	50
CPU ticks	2	10	50	5	5

ALT (exact)	Fl.-E	Fl.-I	Wiki	DBLP	Y!IM
Method	Ikeda	Ikeda	Ikeda	Ikeda	Ikeda
Landmarks used	8	4	4	8	4
Nodes visited	7245	10337	19616	2458	2162
Operations	56502	41349	78647	19666	8648
CPU ticks	7062	10519	25868	1536	1856

acknowledgements

Paolo Boldi Charalampos Tsourakakis

references

Alon, N., Matias, Y., and Szegedy, M. (1999).
The space complexity of approximating the frequency moments.
J. Comput. Syst. Sci., 58(1):137–147.

Alon, U. (2007).
Network motifs: theory and experimental approaches.
Nature Reviews Genetics.

Backstrom, L., Boldi, P., Rosa, M., Ugander, J., and Vigna, S. (2011).
Four degrees of separation.
CoRR, abs/1111.4570.

Boldi, P., Rosa, M., and Vigna, S. (2011).
HyperANF: approximating the neighborhood function of very large graphs on a budget.
In *WWW*.

Bordino, I., Donato, D., Gionis, A., and Leonardi, S. (2008).
Mining large networks with subgraph counting.
In *ICDM*.

references (cont.)

Buriol, L. S., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., and Sohler, C. (2006).
Counting triangles in data streams.
In *PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 253–262, New York, NY, USA. ACM Press.

Das Sarma, A., Gollapudi, S., Najork, M., and Panigrahy, R. (2010).
A sketch-based distance oracle for web-scale graphs.
In *WSDM*, pages 401–410.

Feigenbaum, J., Kannan, S., Gregor, M. A., Suri, S., and Zhang, J. (2004).
On graph problems in a semi-streaming model.
In *31st International Colloquium on Automata, Languages and Programming*.

Flajolet, F., Fusy, E., Gandouet, O., and Meunier, F. (2007).
Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm.
In *Proceedings of the 13th conference on analysis of algorithm (AofA)*.

references (cont.)

Flajolet, P. and Martin, N. G. (1985).
Probabilistic counting algorithms for data base applications.
Journal of Computer and System Sciences, 31(2):182–209.

Goldberg, A. and Harrelson, C. (2005).
Computing the shortest path: A* search meets graph.
In *SODA*.

Kang, U., Tsourakakis, C. E., Appel, A. P., Faloutsos, C., and Leskovec, J. (2011).
HADI: Mining radii of large graphs.
ACM TKDD, 5.

Kleinberg, J., Slivkins, A., and Wexler, T. (2004).
Triangulation and embedding using small sets of beacons.
In *FOCS*.

Ng, E. and Zhang, H. (2008).
Predicting internet network distance with coordinate-based approaches.
In *INFOCOMM*.

references (cont.)

Palmer, C. R., Gibbons, P. B., and Faloutsos, C. (2002).
ANF: a fast and scalable tool for data mining in massive graphs.
In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 81–90, New York, NY, USA. ACM Press.

Thorup, M. and Zwick, U. (2005).
Approximate distance oracles.
JACM, 52(1):1–24.

Tsourakakis, C., Kolountzakis, M., and Miller, G. (2011).
Triangle sparsifiers.
Journal of Graph Algorithms and Applications, 15(6).