

# Graph Analysis in the Wild: Theory to Practice

C. Seshadhri  
(Seshadhri Comandur)

Dept of Computer Science

# Theory and practice

There and back again



## Theory

(CS Theory)

Algorithms  
Graph theory  
Probability theory  
Combinatorics

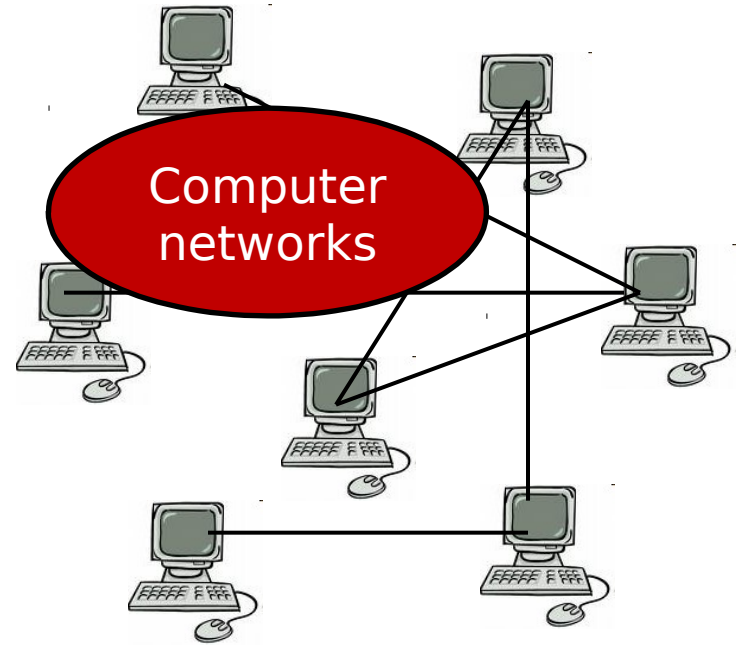
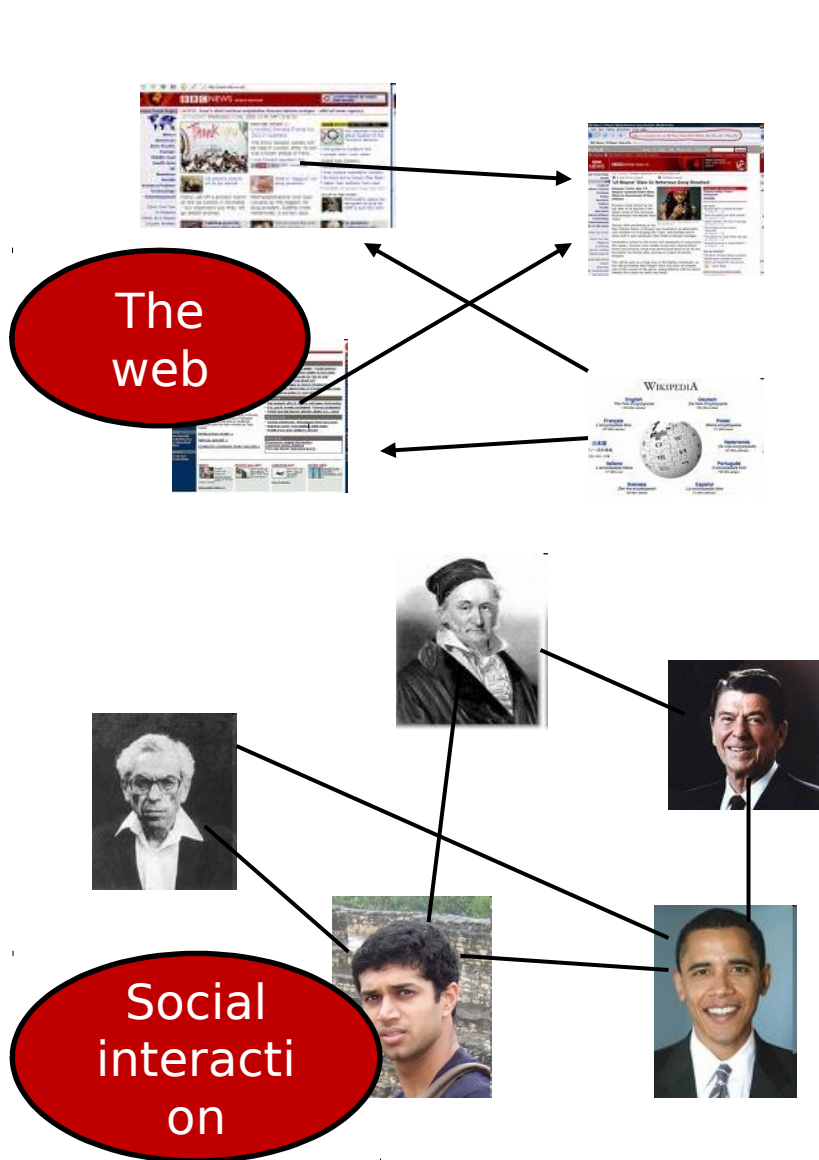


## Practice

(Graph analysis)

Graph modeling  
Clustering  
Pattern counting  
Scalable algorithms

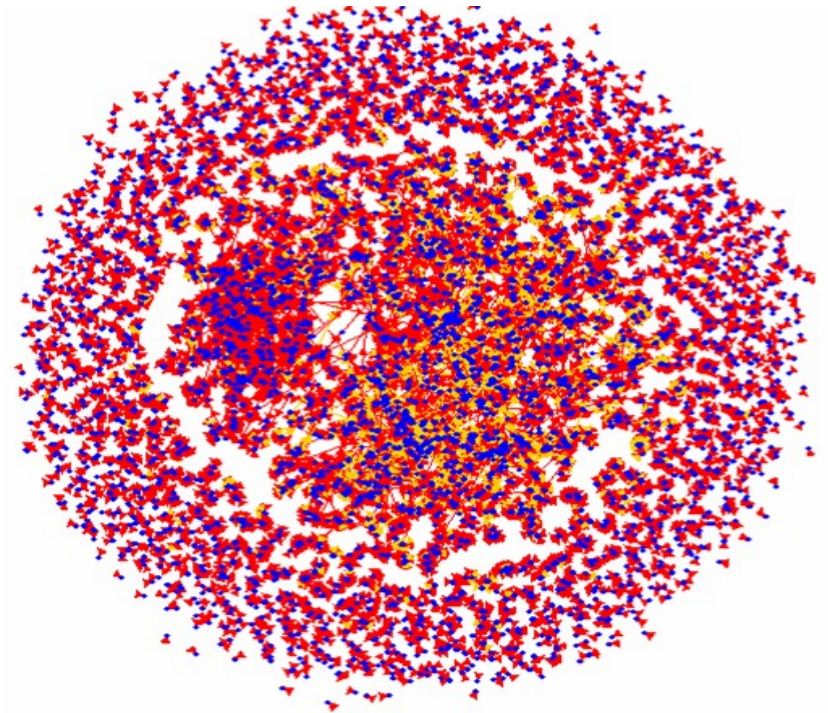
# Large graphs



- Convenient representation
- Vertices and edges connecting them
- Big, millions, billions,...

# So what about them?

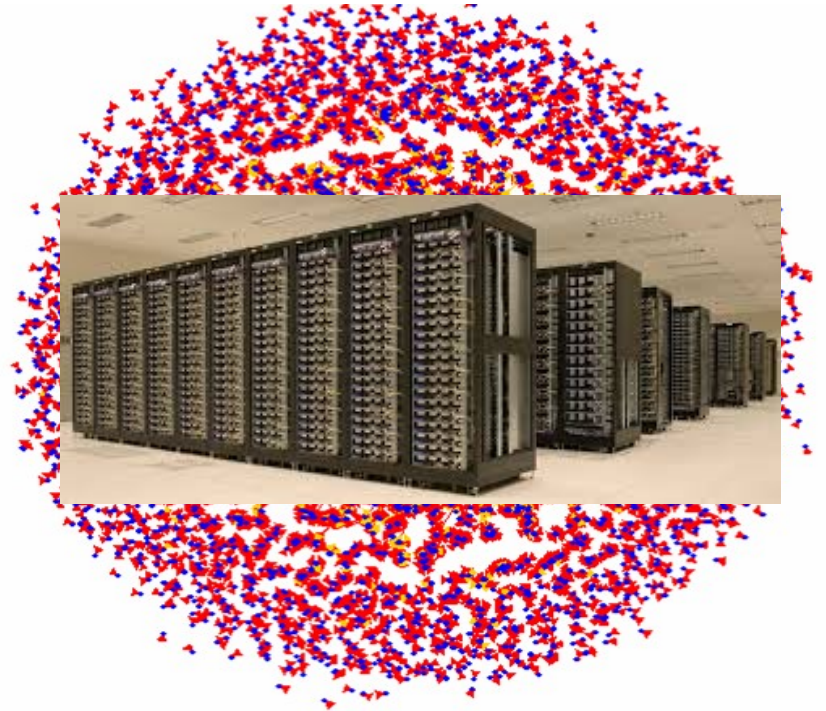
- Are there common patterns?
- Can you find “interesting” activity?
- Is this graph “special”?
- Which regions should I “care” about?



Often reduces to some algorithmic question  
(via graph theory)

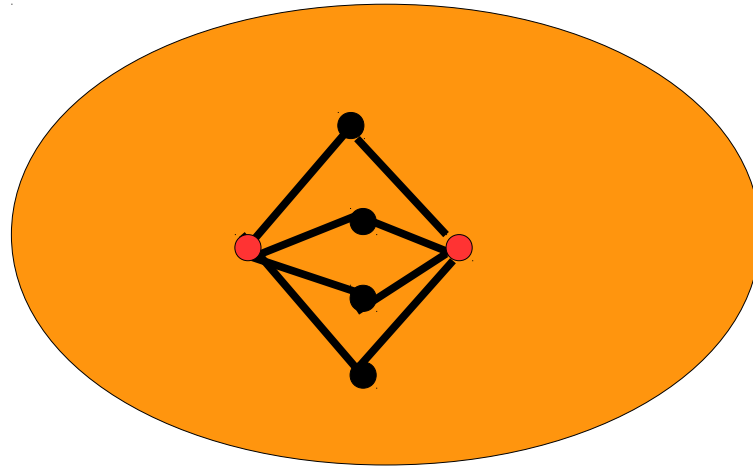
# The size problem

- Graph is massive (millions to billions)
- Distributed
- Noisy
- Probably can't see of all it
- Maybe only one pass of data



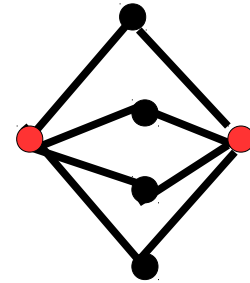
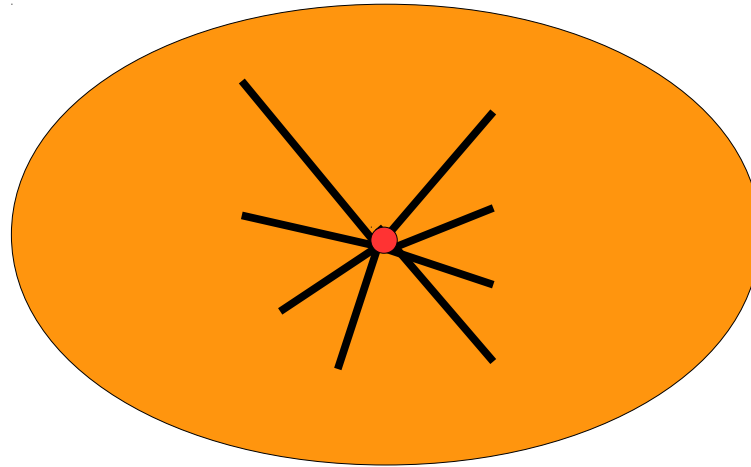
We need to rethink the past 50 years of graph algorithmics.

# Most common neighbors



- $G$  is undirected, simple graph
- Find top  $k$  pairs with most nearest neighbors

# Some naïve algorithms

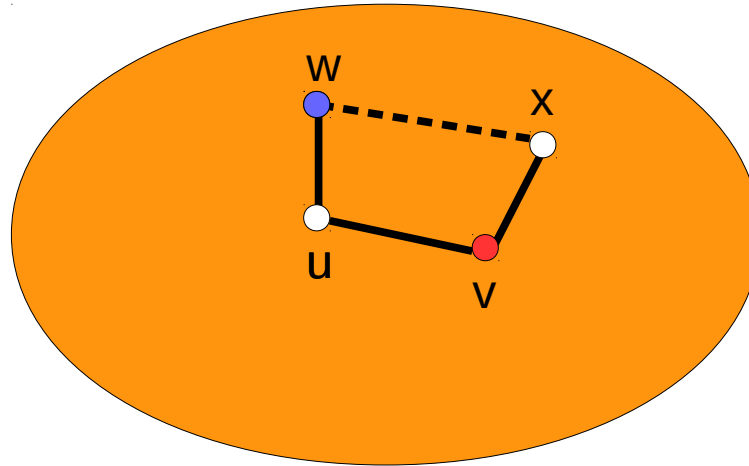


- Try all pairs
  - Complexity  $n^2 \sim 10^{12}$
- Do 2-step BFS from each node
  - Complexity superlinear  $\sim 10^9$



# What we did

[Ballard Pinar Kolda S ICDM15]



Repeat 10,000 times

Output pairs that are frequently reported

- Pick  $e = (u,v)$  with probability prop. to  $d_u d_v$
- Pick random neighbor  $w$  of  $u$ , and  $x$  of  $v$
- Check if  $(w,x)$  exists
- If so, output  $u,w$  as candidate



# Why would you do that?

- Works well (68M edge graph, top 1000 pairs in a second)

where we assume  $t' \ll mn$ . We let  $\Omega_s$  denote the indices of all the nonzeros in  $X$  and  $\Omega_{t'}$  denote the top- $t'$  entries in  $X$ ; this requires a sort in [Line 1](#) of at most  $s$  items (and generally many fewer, depending on the proportion of three-paths that close into diamonds). We compute the  $t'$  dot products in [Lines 3 to 5](#) at a cost of  $O(t'd)$ . Finally, we let  $\Omega_t$  denote the top- $t$  dot products from  $\Omega_{t'}$  in [Line 6](#), requiring a sort of  $t'$  items.

#### Algorithm 2 Postprocessing

Given  $\Omega_s = \{(i, j) \mid x_{ij} > 0\}$ . Let  $t$  be the number of top dot products, and  $t' \geq t$  be the budget of dot products.

- 1: Extract top- $t'$  entries of  $X$ , i.e.,  $|\Omega_{t'}| \leq t'$  and  $\Omega_{t'} \leftarrow \{(i, j) \in \Omega_s \mid x_{ij} \geq x_{i'j'} \forall (i', j') \in \Omega_s \setminus \Omega_{t'}\}$
- 2:  $C \leftarrow$  all-zero matrix of size  $m \times n$
- 3: **for**  $(i, j) \in \Omega_{t'}$  **do**
- 4:  $c_{ij} \leftarrow a_i^T b_j$
- 5: **end for**
- 6: Extract top- $t$  entries of  $C$ , i.e.,  $|\Omega_t| \leq t$  and  $\Omega_t \leftarrow \{(i, j) \in \Omega_{t'} \mid c_{ij} \geq c_{i'j'} \forall (i', j') \in \Omega_{t'} \setminus \Omega_t\}$

#### B. General inputs

We present the binary version as general motivation, but our implementation and analysis are based on the diamond sampling algorithm for general real-valued  $A$  and  $B$  in [Algorithm 3](#). In this case, we define the matrix of weights  $W \in \mathbb{R}^{d \times n}$  such that

$$w_{ki} = |a_{ki}| \|a_{*i}\|_1 \|b_{k*}\|_1 \quad \text{for all } k \in [d], i \in [m].$$

The weight  $w_{ki}$  correspond to the weight of all three paths with edge  $(i, k)$  at its center. This is computed in [Line 2](#). The sampling in [Line 6](#) has the same complexity as in the binary case, but the sampling in [Lines 7 and 8](#) now has a nonuniform distribution and so has higher complexity than in the binary case. The postprocessing is unchanged.

#### Algorithm 3 Diamond sampling with general inputs

Given matrices  $A \in \mathbb{R}^{m \times d}$  and  $B \in \mathbb{R}^{n \times d}$ .

Let  $s$  be the number of samples.

- 1: **for all**  $a_{ki} \neq 0$  **do**
- 2:  $w_{ki} \leftarrow |a_{ki}| \|a_{*i}\|_1 \|b_{k*}\|_1$
- 3: **end for**
- 4:  $X \leftarrow$  all-zero matrix of size  $m \times n$
- 5: **for**  $\ell = 1, \dots, s$  **do**
- 6: Sample  $(k, i)$  with probability  $w_{ki}/\|W\|_1$
- 7: Sample  $j$  with probability  $|b_{kj}|/\|b_{k*}\|_1$
- 8: Sample  $k'$  with probability  $|a_{k'i}|/\|a_{*i}\|_1$
- 9:  $x_{ij} \leftarrow x_{ij} + \text{sgn}(a_{ki}b_{kj}a_{k'i}) b_{k'j}$
- 10: **end for**
- 11: Postprocessing (see [Algorithm 2](#))

1) *Nonnegative inputs*: If  $A$  and  $B$  are nonnegative, the only change is that the sign computations can be ignored in computing the sample increment in [Line 9](#) in [Algorithm 3](#). This avoids potentially expensive random memory accesses.

2) *Equal inputs (Gram matrix)*: If  $B = A$ , then  $C =$  is symmetric. The matrix  $X$  is not symmetric, although it is. Hence, we modify  $X$  before by inserting the following before the postprocessing in [Line 11](#) in [Algorithm 3](#):

$$X \leftarrow (X + X^T)/2.$$

Now  $X$  is symmetric, and the forthcoming analysis is fected.

3) *Equal symmetric inputs (squared matrix)*: If  $B = A$  is symmetric, then  $C = A^2$  and we can replace [Line 3](#) in [Algorithm 3](#) with the following two lines:

$$\begin{aligned} x_{ij} &\leftarrow x_{ij} + \text{sgn}(a_{ki}b_{kj}a_{k'i}) b_{k'j}/2, \\ x_{k'k} &\leftarrow x_{k'k} + \text{sgn}(a_{ki}b_{kj}a_{k'i}) b_{k'j}/2. \end{aligned}$$

This exploits the fact that we can swap the role of  $k$  and the initial edge sample. Again,  $X$  may not be symmetric we insert (2) before the postprocessing in [Line 11](#).

#### C. Complexity and space

Let  $\alpha = \text{nnz}(A)$  and  $\beta = \text{nnz}(B)$ . In the dense case,  $md$  and  $\beta = nd$ . The total work is

$$O(\alpha + \beta + s \log(\alpha\beta)).$$

The total storage (not counting the inputs  $A$  and  $B$ ) is

$$2 \text{storage}(A) + \text{storage}(B) + 5s + 3t' + 3t.$$

We give detailed arguments below and in the implementation discussion in [Section V](#).

**Preprocessing.** For the sampling in [Lines 7 and 8](#) precompute cumulative, normalized column sums for  $A$  the same for rows of  $A$ , requiring storage of  $\text{storage}(A)$  and computation of  $O(\alpha + \beta)$ . The matrix  $W$  the same nonzero pattern as  $A$ , so the cost to store it is to  $\text{storage}(A)$  and to compute it is  $O(\alpha)$ .

**Sampling.** For a straightforward implementation, the per sample in [Line 6](#) is  $O(\log(\alpha))$ . For [Line 7](#), the cost sample is  $O(\log(\beta/d))$ ; here, we have used the approximation  $\text{nnz}(b_{k*}) \approx \beta/d$ . A similar analysis applied for  $A$  and  $B$ . So, the cost per sample is  $O(\log(\alpha) + \log(\beta/d) + \log(\alpha))$ . Without loss of generality, we assume that we need to the three-paths and the summand in [Line 9](#) for a total of  $5s$ .

**Postprocessing.** Conservatively, we require  $3t'$  storage for the  $(i, j, x_{ij} \text{ or } c_{ij})$  triples in  $\Omega_{t'}$  and  $3t$  storage for  $(i, j, c_{ij})$  triples in  $\Omega_t$ . The sorting requires at most  $O(s)$  time, and usually much less since  $\text{nnz}(X)$  may be much than  $s$  due to only some three-paths forming diamond concentration, i.e., picking the same  $(i, j)$  pair multiple times.

#### IV. ANALYSIS OF DIAMOND SAMPLING

This section provides a theoretical analysis of diamond sampling. We first prove that the expected value of  $\sum c_{ij}^2/\|W\|_1$ , and then we prove error bounds on our estimate as a function of the number of samples. Unless stated otherwise our analysis applies to the general version of the diamond sampling algorithm ([Algorithm 3](#)).

#### A. Expectation

For a single instance of [Lines 6 to 8](#) of [Algorithm 3](#), we define the event

$$\mathcal{E}_{k'ikj} = \text{choosing three-path } (k', i, k, j).$$

LEMMA 1.  $\Pr(\mathcal{E}_{k'ikj}) = |a_{ki}b_{kj}a_{k'i}|/\|W\|_1$ .

*Proof:* The probability of choosing three-path  $(k', i, k, j)$  is (by independence of these choices) the product of the following probabilities: that of choosing the center edge  $(i, k)$ , then picking  $j$ , and then picking  $k'$ .

$$\begin{aligned} \Pr(\mathcal{E}_{k'ikj}) &= \Pr(\text{ctr}(i, k)) \cdot \Pr(\text{endpts } j \text{ and } k' | \text{ctr}(i, k)) \\ &= \frac{w_{ki}}{\|W\|_1} \cdot \frac{|b_{kj}|}{\|b_{k*}\|_1} \cdot \frac{|a_{k'i}|}{\|a_{*i}\|_1} \\ &= \frac{|a_{ki}| |a_{*i}| |b_{k*}|_1}{\|W\|_1} \cdot \frac{|b_{kj}|}{\|b_{k*}\|_1} \cdot \frac{|a_{k'i}|}{\|a_{*i}\|_1} \\ &= \frac{|a_{ki}b_{kj}a_{k'i}|}{\|W\|_1}. \end{aligned}$$

In what follows, we use  $X_{i,j,\ell}$  to be the following random variable: if  $i, j$  are the respective indices updated in the  $\ell$ th iteration,  $X_{i,j,\ell} = \text{sgn}(a_{ki}b_{kj}a_{k'i})b_{k'j}$ . Otherwise,  $X_{i,j,\ell} = 0$ . Observe that  $x_{ij} = \sum_{\ell=1}^s X_{i,j,\ell}$ .

LEMMA 2. For diamond sampling,  $\mathbb{E}[x_{ij}/s] = c_{ij}^2/\|W\|_1$ .

*Proof:* We note that  $\mathbb{E}[x_{ij}/s] = \mathbb{E}[\sum_{\ell} X_{i,j,\ell}]/s = \mathbb{E}[X_{i,j,1}]$ . (We use linearity of expectation and the fact that the  $X_{i,j,\ell}$  are i.i.d. for fixed  $i, j$  and varying  $\ell$ .)

$$\begin{aligned} \mathbb{E}[X_{i,j,1}] &= \sum_{k, k'} \Pr(\mathcal{E}_{k'ikj}) \cdot \text{sgn}(a_{ki}b_{kj}a_{k'i}) b_{k'j} \\ &= \sum_{k, k'} \frac{|a_{ki}b_{kj}a_{k'i}|}{\|W\|_1} \cdot \text{sgn}(a_{ki}b_{kj}a_{k'i}) b_{k'j} \\ &= \frac{1}{\|W\|_1} \sum_{k, k'} a_{ki}b_{kj}a_{k'i}b_{k'j} \\ &= \frac{1}{\|W\|_1} \left( \sum_k a_{ki}b_{kj} \right) \left( \sum_{k'} a_{k'i}b_{k'j} \right) \\ &= \frac{1}{\|W\|_1} \left( \sum_k a_{ki}b_{kj} \right)^2 = \frac{c_{ij}^2}{\|W\|_1}. \end{aligned}$$

#### B. Concentration bounds

We now provide some concentration bounds when all entries in  $A$  and  $B$  are nonnegative.

LEMMA 3. Fix  $c > 0$  and error probability  $\delta \in (0, 1)$ . Assume all entries in  $A$  and  $B$  are nonnegative and at most  $K$ . If the number of samples

$$s \geq 3K \|W\|_1 \log(2/\delta) / (c^2 c_{ij}^2),$$

then

$$\Pr[|x_{ij}/W\|_1/s - c_{ij}^2| > c c_{ij}^2] \leq \delta.$$

*Proof:* Observe that  $X_{i,j,\ell}$  is in the range  $[0, K]$ . Thus,  $Y_{i,j,\ell} = X_{i,j,\ell}/K$  is in  $[0, 1]$ . Set  $y_{ij} = \sum_{\ell} Y_{i,j,\ell}$ . Since  $y_{ij}$  is the sum of random variables in  $[0, 1]$ , we can apply the standard multiplicative Chernoff bound ([Theorem 1.1](#) of [\[32\]](#)). This yields  $\Pr[y_{ij} \geq (1 + \epsilon)\mathbb{E}[y_{ij}]] < \exp(-\epsilon^2\mathbb{E}[y_{ij}]/3)$ . By [Lemma 2](#),  $\mathbb{E}[y_{ij}] = (s/K)(c_{ij}^2/\|W\|_1)$ , which is at least  $3 \log(2/\delta)/\epsilon^2$  by choice of  $s$ . Hence,  $\Pr[y_{ij} \geq (1 + \epsilon)\mathbb{E}[y_{ij}]] < \delta/2$ . Note that  $y_{ij} = x_{ij}/K$ . We multiply the expression inside the  $\Pr[\cdot]$  by  $K\|W\|_1/s$  to get the event  $x_{ij}/W\|_1/s \geq (1 + \epsilon)c_{ij}^2$ .

Using the Chernoff lower tail bound and identical reasoning, we get  $\Pr[x_{ij}/W\|_1/s \leq (1 - \epsilon)c_{ij}^2] \leq \delta/2$ . A union bound completes the proof. ■

The following theorem gives a bound on the number of samples required to distinguish “large” dot products from “small” ones. The constant 4 that appears is mostly out of convenience; it can be replaced with anything  $> 1$  with appropriate modifications to  $s$ .

THEOREM 4. Fix some threshold  $\tau$  and error probability  $\delta \in (0, 1)$ . Assume all entries in  $A$  and  $B$  are nonnegative and at most  $K$ . Suppose  $s \geq 12K\|W\|_1 \log(2mn/\delta)/\tau^2$ . Then with probability at least  $1 - \delta$ , the following holds for all indices  $i, j$  and  $i', j'$ : if  $c_{ij} > \tau$  and  $c_{i'j'} < \tau/4$ , then  $x_{ij} > x_{i'j'}$ .

*Proof:* First consider some dot product  $c_{ij}$  with value at least  $\tau$ . We can apply [Lemma 3](#) with  $\epsilon = 1/2$  and error probability  $\delta/mn$ , so with probability at least  $1 - \delta/mn$ ,  $x_{ij}/W\|_1/s \geq c_{ij}^2/2 \geq \tau^2/2$ . Now consider dot product  $c_{i'j'}$  with  $c_{i'j'} < \tau/3$ . Define  $y_{i'j'}$  and  $Y_{i'j',\ell}$  as in the proof of [Lemma 3](#). We can apply the lower tail bound of [Theorem 1.1](#) (third part) of [\[32\]: for any  \$b > 2e\mathbb{E}\[y\_{i'j'}\]\$ ,  \$\Pr\[y\_{i'j'} > b\] < 2^{-b}\$ .](#)

We set  $b = s\tau^2/2K\|W\|_1$ . From [Lemma 2](#) and the assumption that  $c_{i'j'} < \tau/3$  and  $\mathbb{E}[y_{i'j'}] = \mathbb{E}[x_{i'j'}/K] = sc_{i'j'}^2/K\|W\|_1 \leq s\tau^2/(16K\|W\|_1) < b/2e$ . Plugging in our bound for  $s$ ,  $b \geq 12K\|W\|_1 \log(2mn/\delta)/\tau^2 \cdot \tau^2/(2K\|W\|_1) = 6 \log(2mn/\delta)$ . Hence,  $\Pr[y_{i'j'} > b] < \delta/(2mn)$ . Equivalently,  $\Pr[x_{i'j'}/W\|_1/s > \tau^2/2] < \delta/(2mn)$ . We take a union bound over all the error probabilities (there are at most  $mn$  pairs  $i, j$  or  $i', j'$ ).

In conclusion, with probability at least  $1 - \delta$ , for any pair of indices  $i, j$ : if  $c_{ij} > \tau$ , then  $x_{ij}/W\|_1/s \geq \tau^2/2$ . If  $c_{i'j'} < \tau/4$ , then  $x_{i'j'}/W\|_1/s < \tau^2/2$ . This completes the proof. ■

To get a useful interpretation of [Lemma 3](#) and [Theorem 4](#), we ignore the parameters  $\epsilon$  and  $\delta$ . Let us also assume that  $K = 1$ , which is a reasonable assumption for most of our experiments. Basically, to get a reasonable estimate of  $c_{ij}$ , we require  $\|W\|_1/c_{ij}^2$  samples. If the value of the  $t$ -th largest entry in  $C$  is  $\tau$ , we require  $\|W\|_1/\tau^2$  samples to find the  $t$ -largest entries. For instance, on a graph, if we want to identify pairs of vertices with at least 200 common neighbors, we can set  $\tau = 200$ , and  $\|W\|_1$  will be the number of (non-induced) 3-paths in the graph. The square in the denominator is what makes this approach work. In [Table I](#) of [Section VI](#), we show some of the values of  $\|W\|_1/\tau^2$  for particular datasets, where  $\tau$  is the magnitude of the largest entry.

# My research



Simple, practical  
graph problem



Use theory  
for algorithm

Apply algorithm  
in practice

Prove it's  
correct!

Show it's  
useful!

# What theory?



Graph  
Theory

Information  
Theory

$$H = -\sum p_i \log p_i$$

Combinatorics



**Theoretical  
computer science**

Probability  
Theory



$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

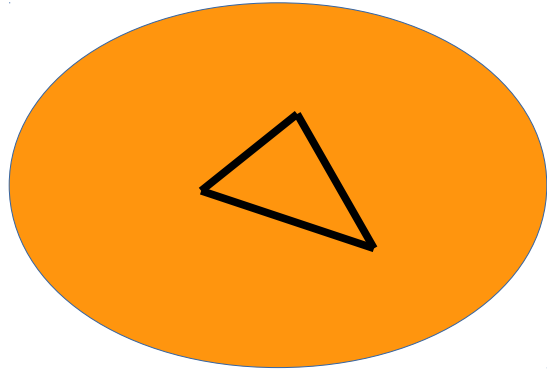
$$f(t) = \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega$$

Functional  
Analysis

Linear  
Algebra

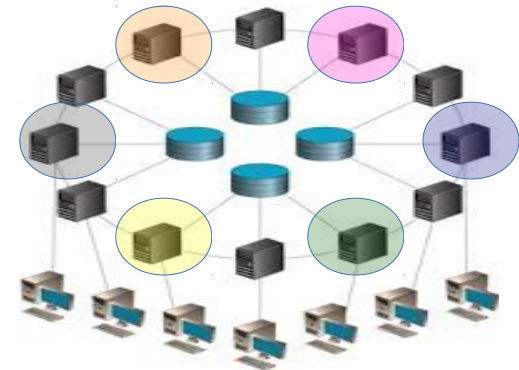
$$Ax = b$$

# Triangle counting

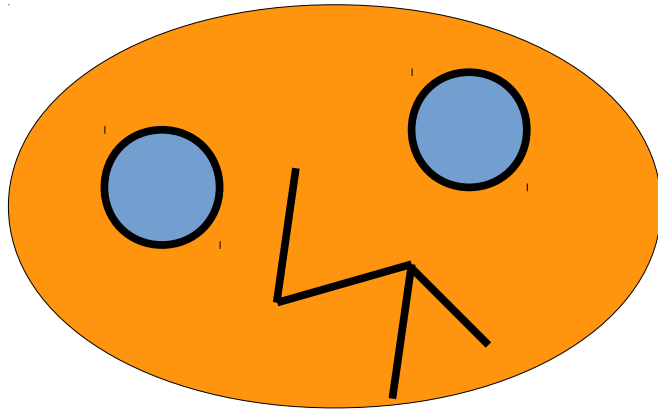


Sociology, physics, bioinformatics,  
graph clustering, machine learning,...

Every situation has led to new  
algorithms, new math, and new  
applications



# Theoretical foundations



```
AACTCAAGCTTCTCTGTGTGGCTGTGTGATCTGGCTCAAAAGCCACAGCCCTGGCTTAGCAGC
AGGACCTTGGATGGTCTGTCGACAGATGAGGAGAATCTCTCTTTTCTGCTGGCTTGAAG
GACAGACATGACTTTGGATTTCCCGAGGAGGAGTTTGGCCAAACAGTTCCAAAAGGCT
GAAACCATCCCTGTCTCTGATGAGGATGATCCAGCCAGATCTTCAATCTCTTCAGCCACA
AAGGACTCATCTGGCTGTGGGATGAGACCTCTCAGACAAATTTCTAGACTGAACTC
TACCAGCAGCTGAAATGACCTTGGGANGCTGTCTGATACAGGCGGTGGCGGTGACAGAG
ACTCCCTGATGAAAGGAGGACTCCATCTCTGGCTGTGAGGAAATACTTCCAAAAGAACTC
ACTCTCTATCTGAAAGAGAAAGAAATACAGCCCTTGTGGCTTGGGAGTTGTTCAGAGCCA
GAAATCATGAGACTTTTTCTTTTCTCAAAACTTCCAGAAAGTTTAAAGAGTAAAG
GAATGA, TGTGATCTGGCTCAAAAGCCACAGCCCTGGCTTAGCAGGAGGAGCCCTGATCC
TCTTGGCAGAGATGAGGAGAAATCTCTCTTTTCTCTCTGCTTGAAGGACAGACATGACT
TTGGATTTCCCGAGGAGGTTTGGCCAAACAGTTCCAAAAGTTGAAACCATCCCTG
TCTTCCATGATGATGATCCAGCCAGATCTTCAATCTCTTCCAGCCAAAGGACTCATCTG
CTGCTTGGGATGAGACCTCTAGACAAATTTCTAGACTGAACTCTACAGCAGCTGA
ATGACCTGGAAGCCCTGTGTGATACAGGCGGTGGCTGGACAGTACTCCCTGTATGA
AGGAGACTCCATCTCTGGCTGTGAGGAAATACTTCCAAAAGAAATCTCTCTATCTGA
AAGGAGAGAAAACAGCCCTTGTGGCTTGGGAGGTTGTTCAGGACAGAAATCATGAT
CTTTTCTTTTGTCAAGAACTTGGCAGAAAGTTTAAAGAGTAAAGGAAATGA and
```

Sublinear algorithms: How to sample a large “function” for algorithmic purposes?

PhD sutra

Your advantage over your  
advisor is **time**



Don't just read it. Fight it

- P. R. Halmos

A paper is a map. Maps are used  
for traveling, not reading

**Go back to the big picture**

Everyone has slumps

Learn to present

There are no boundaries  
between fields

Ask questions