SAND2015-0782C

Sandia National Laboratories

Computing the Largest Entries in a Matrix Product via Sampling

Grey Ballard, *Tamara G. Kolda*, Ali Pinar (Sandia National Labs)
and C. Seshadhri (UC Santa Cruz)

U.S. Department of Defense
Defense Advanced Research Projects Agency

U.S. DEPARTMENT OF ENERGY

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.


DON'T WORRY, IT'S GONNA BE COOL... HOMER'S ON THE JOB, MAN!
THIS SITE UNDER CONSTRUCTION

Matrices and Graphs

Sandia National Laboratories

Simple Undirected Graph

$V = \text{set of vertices}, \; n = |V|$

$E = \text{set of undirected edges}$

$G = (V, E) = \text{graph}$

Matrices and Graphs

Sandia National Laboratories

Simple Undirected Graph

$V = \text{set of vertices}, \; n = |V|$

$E = \text{set of undirected edges}$

$G = (V, E) = \text{graph}$

Adjacency Matrix

$A = n \times n \text{ symmetric matrix}$

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E, \; i \neq j \\ 0 & \text{otherwise} \end{cases}$$

Matrices and Graphs

Sandia National Laboratories

Simple Undirected Graph

$V = \text{set of vertices}, \; n = |V|$

$E = \text{set of undirected edges}$

$G = (V, E) = \text{graph}$

Adjacency Matrix

$A = n \times n \text{ symmetric matrix}$

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E, \; i \neq j \\ 0 & \text{otherwise} \end{cases}$$

Vertex Degree

$$d_i = \sum_{j=1}^{n} a_{ij} = \text{degree of node } i$$

Matrices and Graphs

Sandia National Laboratories

Simple Undirected Graph

$V = \text{set of vertices}, \; n = |V|$

$E = \text{set of undirected edges}$

$G = (V, E) = \text{graph}$

Adjacency Matrix

$A = n \times n \text{ symmetric matrix}$

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E, \; i \neq j \\ 0 & \text{otherwise} \end{cases}$$

Vertex Degree

$$d_i = \sum_{j=1}^{n} a_{ij} = \text{degree of node } i$$

Cycles and Paths

$P_K = \text{set of paths of length K}$

$C_K = \text{set of cycles of length K}$

Matrix Computations for Graphs

Sandia National Laboratories

Key Well-Known Result

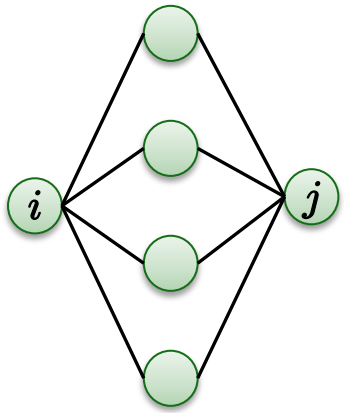
$(A^K)_{ij}$ = number of paths of length K between vertex pair (i,j)

Matrix Computations for Graphs

Key Well-Known Result

$(A^K)_{ij}$ = number of paths of length K between vertex pair (i,j)

$(A^2)_{ij}$ = number common nghbrs for vertex pair (i,j)

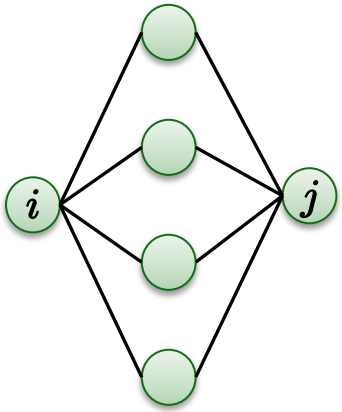


Matrix Computations for Graphs

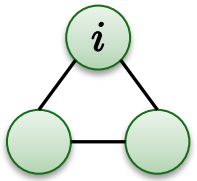
Key Well-Known Result

$(A^K)_{ij}$ = number of paths of length K between vertex pair (i,j)

$(A^2)_{ij}$ = number common nghbrs for vertex pair (i,j)



$$|C_3| = \frac{1}{6}\sum_{i=1}^{n}(A^3)_{ii}$$ = # of triangles



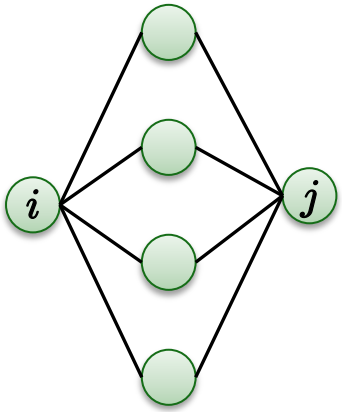
Matrix Computations for Graphs

Sandia National Laboratories

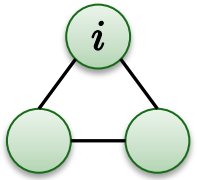
Key Well-Known Result

$(A^K)_{ij}$ = number of paths of length K between vertex pair (i,j)

$(A^2)_{ij}$ = number common nghbrs for vertex pair (i,j)

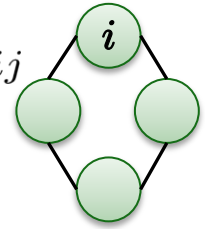


$$|C_3| = \frac{1}{6}\sum_{i=1}^{n}(A^3)_{ii}$$ = # of triangles



$$|C_4| = \frac{1}{8}\sum_{i=1}^{n}(A^4)_{ii} - \frac{1}{4}\sum_{i,j=1}^{n}(A^2)_{ij}$$

= number of 4-cycles



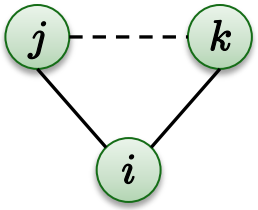
Sandia National Laboratories

Computing matrix products is too expensive in computation and memory!

Counting Triangles Exactly

Sandia National Laboratories

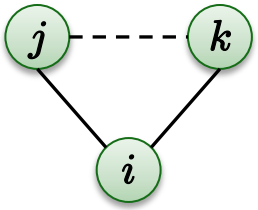
```
1: x ← 0
2: for (j, i, k) ∈ P_2 do
3:     if (j, k) ∈ E then
4:         x ← x + 1
5:     end if
6: end for
```



- Basic idea
  - Check each wedge to see if it forms a triangle
  - Counts each triangle 3X

Counting Triangles Exactly

```
1: x ← 0
2: for (j, i, k) ∈ P2 do
3:     if (j, k) ∈ E then
4:         x ← x + 1
5:     end if
6: end for
```

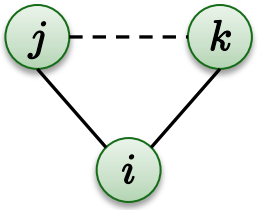


$$|P_2| = \sum_{i=1}^{n} d_i(d_i - 1)/2$$

- Basic idea
  - Check each wedge to see if it forms a triangle
  - Counts each triangle 3X
- Cost
  - Proportional to number of wedges $|P_2|$

Counting Triangles Exactly

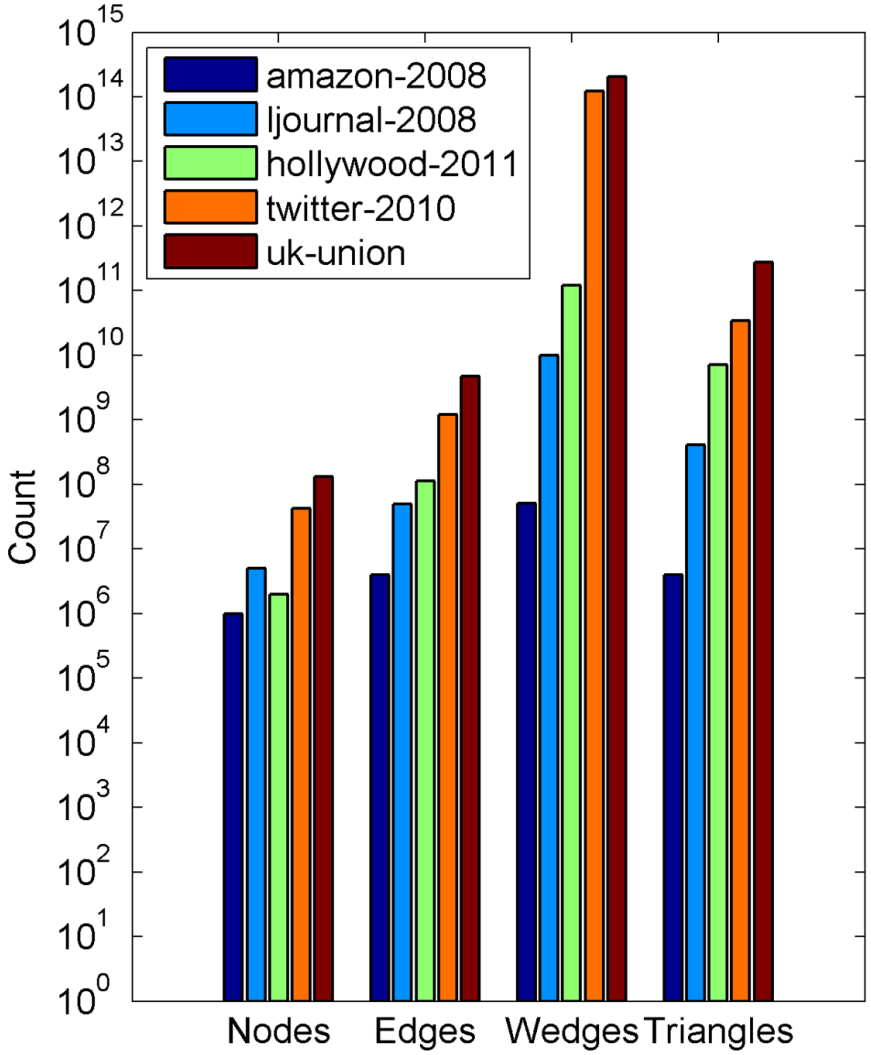
```
1: x ← 0
2: for (j, i, k) ∈ P2 do
3:     if (j, k) ∈ E then
4:         x ← x + 1
5:     end if
6: end for
```



$$|P_2| = \sum_{i=1}^{n} d_i(d_i - 1)/2$$

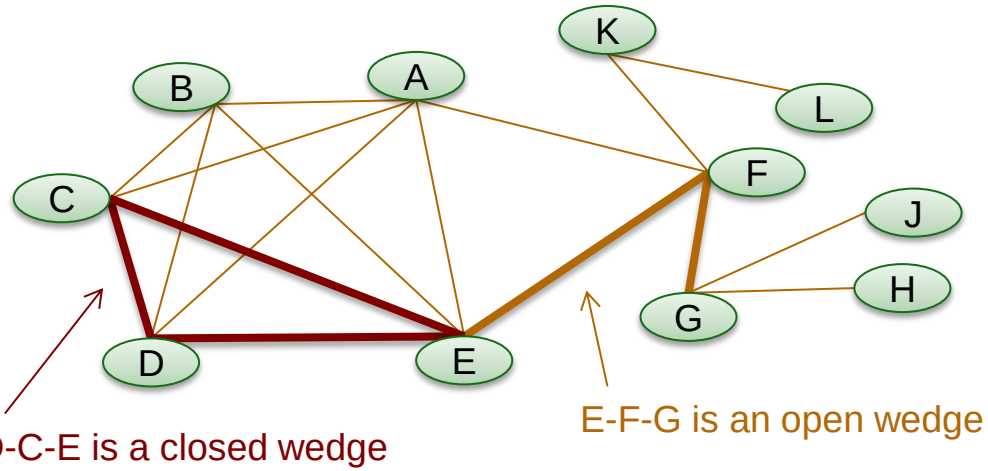
- Basic idea
  - Check each wedge to see if it forms a triangle
  - Counts each triangle 3X
- Cost
  - Proportional to number of wedges $|P_2|$
- Improvements
  - Reject wedges $i > j$ or $i > k$
    - Count each triangle 1X
    - Eliminates many wedges
  - Order nodes by incr. degree
    - Even fewer wedges

Example Wedge & Triangle Counts



Wedge Polling

Define $\tau = \frac{3|C_3|}{|P_2|}$ = proportion of wedges that are *closed*.

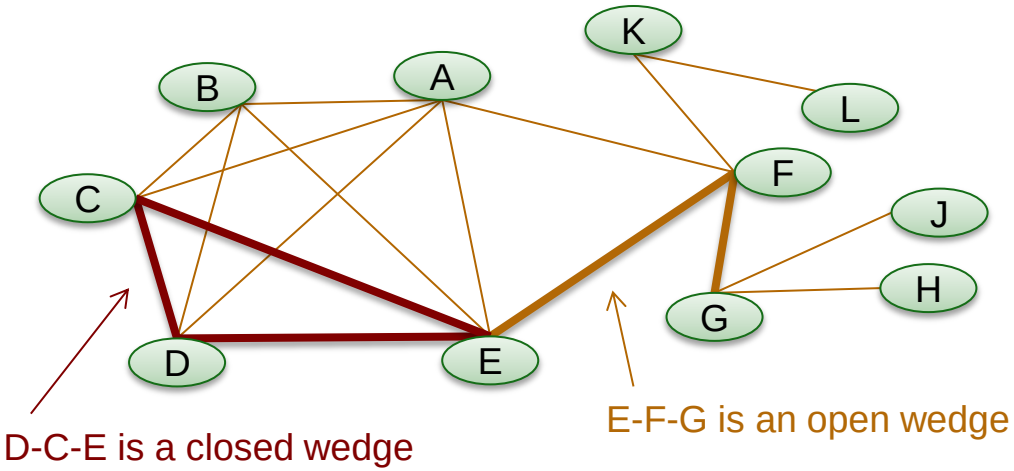


Enumeration: Find *every* wedge. Check if each is closed.
τ = # closed wedges / # wedges

Seshadhri, Pinar, Kolda, *SIAM Intl. Conf. Data Mining,* 2013; Seshadhri, Pinar, Kolda, *Statistical Analysis and Data Mining*, 2014

Wedge Polling

Define $\tau = \frac{3|C_3|}{|P_2|}$ = proportion of wedges that are *closed*.

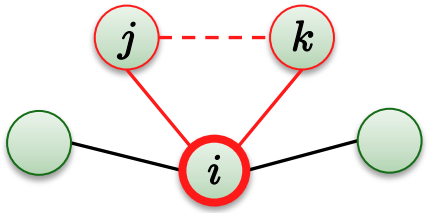


Enumeration: Find *every* wedge. Check if each is closed.
τ = # closed wedges / # wedges

Sampling: Sample a few wedges (uniformly). Check if each is closed.
$\tau \approx$ # closed sampled wedges / # sampled wedges

Seshadhri, Pinar, Kolda, *SIAM Intl. Conf. Data Mining,* 2013; Seshadhri, Pinar, Kolda, *Statistical Analysis and Data Mining*, 2014

Counting Triangles via Sampling

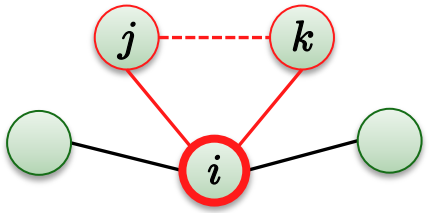


- Basic Idea
  - s = # wedge samples
  - x = # closed sample wedges
  - x/s = estimate of τ
  - Pick uniformly random wedge *without explicit enumeration*

Counting Triangles via Sampling

Algorithm Compute degrees

1: $d \leftarrow 0$
2: **for** $(i, j) \in E$ **do**
3: $\quad d_i \leftarrow d_i + 1,\ d_j \leftarrow d_j + 1$
4: **end for**



- Basic Idea
  - s = # wedge samples
  - x = # closed sample wedges
  - x/s = estimate of τ
  - Pick uniformly random wedge *without explicit enumeration*

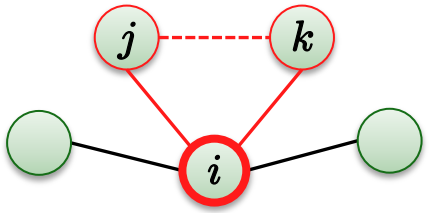
Counting Triangles via Sampling

Algorithm Compute degrees

$d \leftarrow 0$
for $(i,j) \in E$ **do**
    $d_i \leftarrow d_i + 1,\ d_j \leftarrow d_j + 1$
end for

Algorithm Sample

$x \leftarrow 0$
for $\alpha = 1, \ldots, s$ **do**
    Choose $i \propto d_i(d_i - 1)/2$
    Choose $j \in \mathcal{N}(i)$
    Choose $k \in \mathcal{N}(i) \setminus \{j\}$
    if $(j,k) \in E$ **then**
        $x \leftarrow x + 1$
    end if
end for
$x \leftarrow (|P_3|/3) \cdot (x/s)$



- Basic Idea
  - s = # wedge samples
  - x = # closed sample wedges
  - x/s = estimate of τ
  - Pick uniformly random wedge *without explicit enumeration*

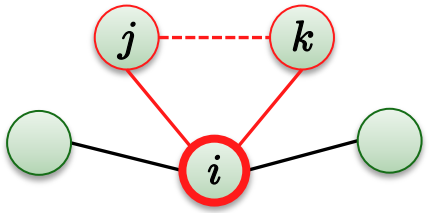
Counting Triangles via Sampling

Algorithm Compute degrees

```
1: d ← 0
2: for (i, j) ∈ E do
3:     d_i ← d_i + 1, d_j ← d_j + 1
4: end for
```

Algorithm Sample

```
1: x ← 0
2: for α = 1, ..., s do
3:     Choose i ∝ d_i(d_i − 1)/2
4:     Choose j ∈ N(i)
5:     Choose k ∈ N(i) \ {j}
6:     if (j, k) ∈ E then
7:         x ← x + 1
8:     end if
9: end for
10: x ← (|P_3|/3) · (x/s)
```



- Basic Idea
  - s = # wedge samples
  - x = # closed sample wedges
  - x/s = estimate of τ
  - Pick uniformly random wedge *without explicit enumeration*
- Cost
  - Preprocess: O($|E|$)
  - Sample: O(s log(n))

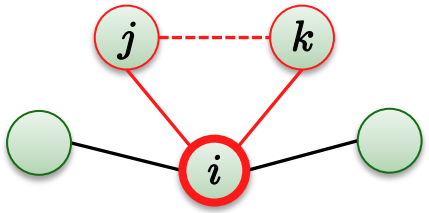
Counting Triangles via Sampling

Algorithm Compute degrees

$d \leftarrow 0$
for $(i,j) \in E$ **do**
  $d_i \leftarrow d_i + 1,\ d_j \leftarrow d_j + 1$
end for

Algorithm Sample

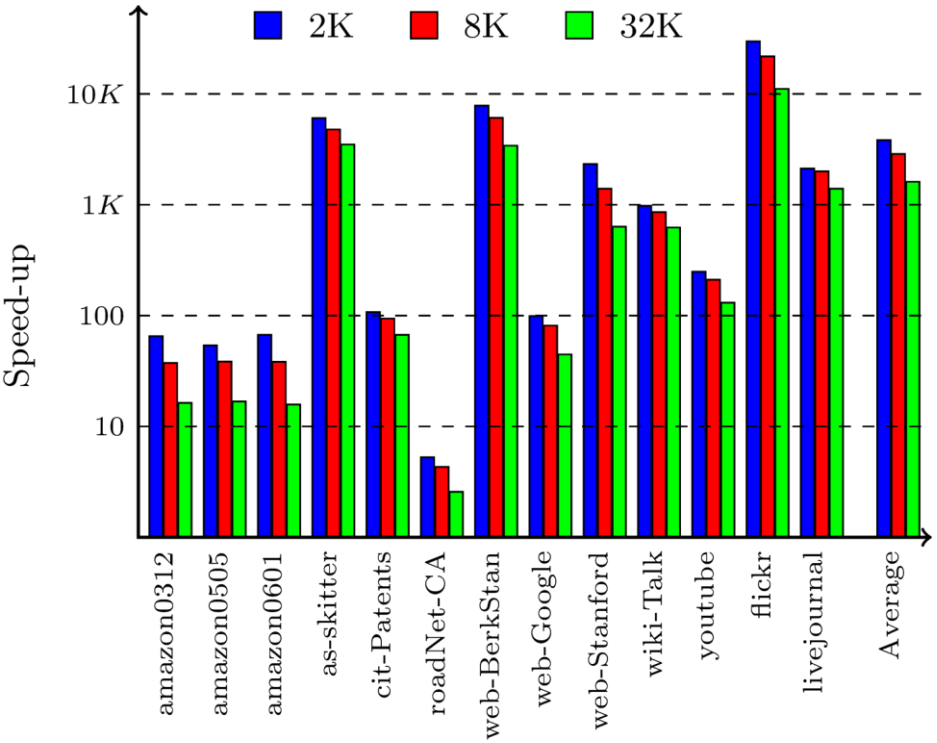
$x \leftarrow 0$
for $\alpha = 1, \ldots, s$ **do**
  Choose $i \propto d_i(d_i - 1)/2$
  Choose $j \in \mathcal{N}(i)$
  Choose $k \in \mathcal{N}(i) \setminus \{j\}$
  if $(j,k) \in E$ **then**
    $x \leftarrow x + 1$
  end if
end for
$x \leftarrow (|P_3|/3) \cdot (x/s)$

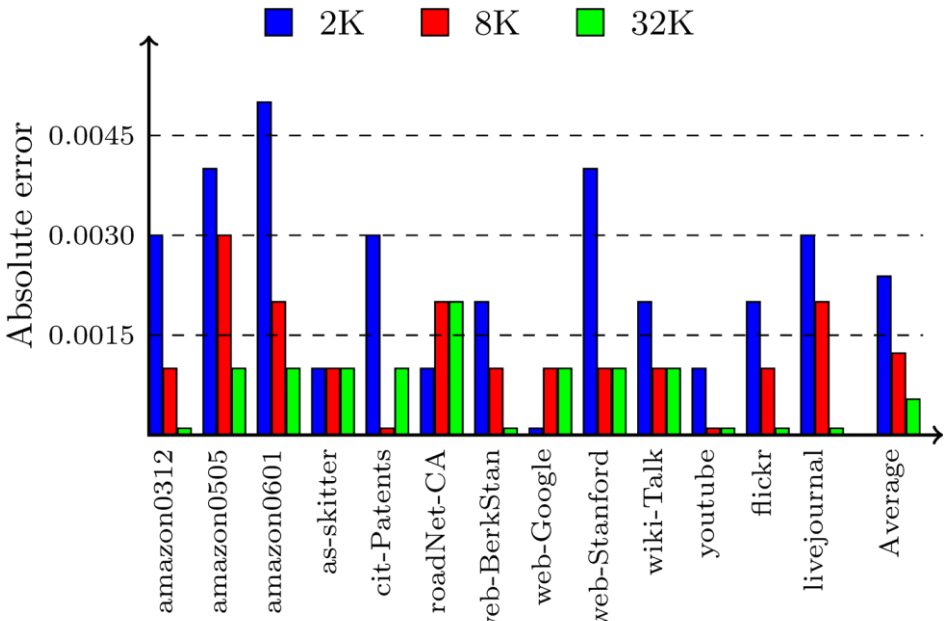


- Basic Idea
  - s = # wedge samples
  - x = # closed sample wedges
  - x/s = estimate of τ
  - Pick uniformly random wedge *without explicit enumeration*
- Cost
  - Preprocess: O($|E|$)
  - Sample: O(s log(n))

Linear with "alias" method

Sampling gives 1000X Speed-up with High Accuracy





Sampling Theory

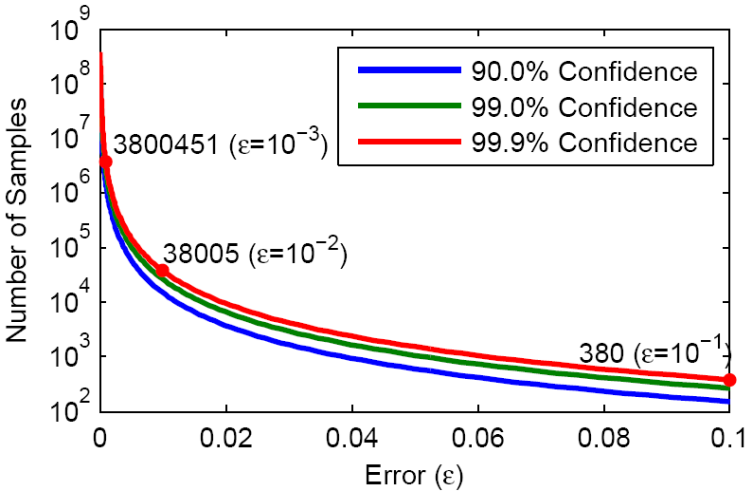
Bounded Error: Hoeffding's Inequality

Theorem: (Hoeffding 1963) Let $X_1, X_2, \ldots, X_k \in [0,1]$ be independent random variables. Define the sample mean: $\bar{X} = \frac{1}{k}\sum_{i=1}^{k} X_i$

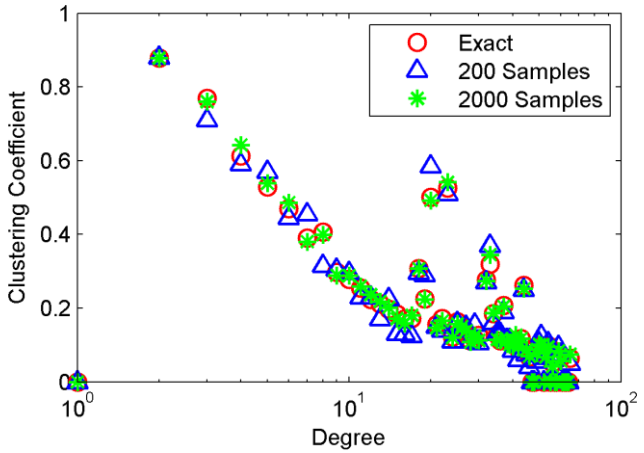
Let μ be the true mean. Then for $\epsilon \in (0, 1-\mu)$,

$$\text{Prob}\left\{|\bar{X} - \mu| \geq \epsilon\right\} \leq \delta \equiv 2\exp(-2k\epsilon^2)$$

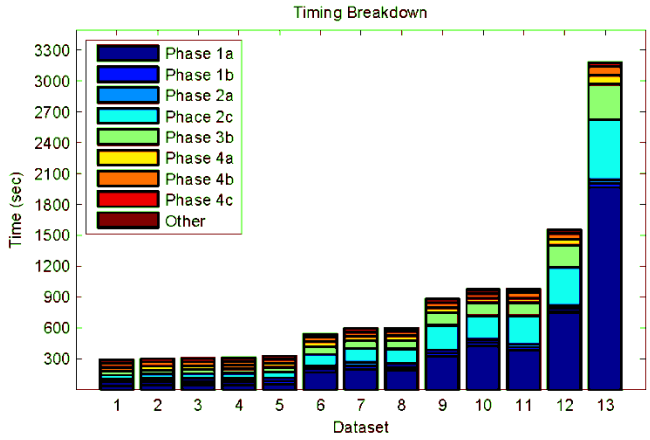
Hence, for a given error ϵ and confidence 1-δ, we just need to set $k = \lceil 0.5\epsilon^{-2}\log(2/\delta)\rceil$



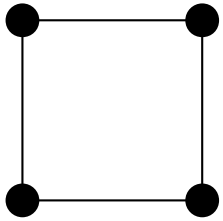
Degree-wise Computation



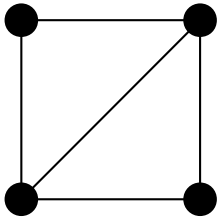
We also have a MapReduce implementation: Kolda, Pinar, Plantenga, Seshadhri, Task, SISC 2014.



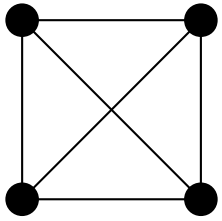
Counting 4-Cycles



C_4

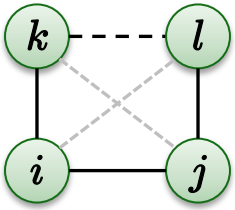


C_4+chord



K_4

- Basic idea
  - For each 3-path (k,i,j,l), if edge (k,l) exists, increment count appropriately

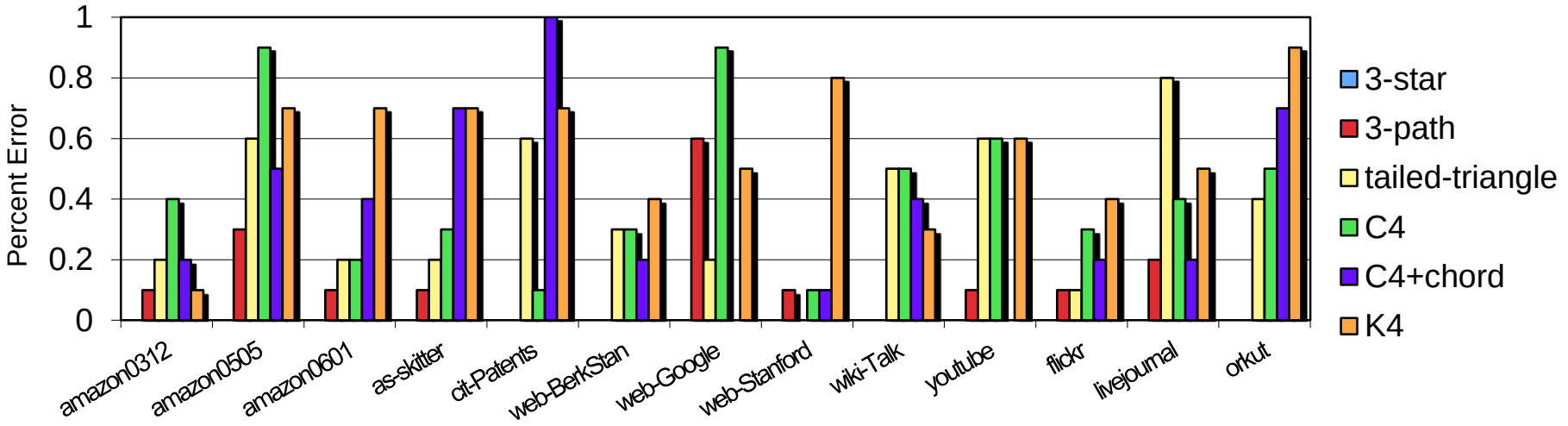


  - *Each 4-cycle is counted multiple times, depending on 4-vertex pattern*
  - Reduce work by just considering "centered 3-paths"

Jha, Seshadhri, Pinar, *WWW* '15

New 3-path sampling algorithm is fast and accurate

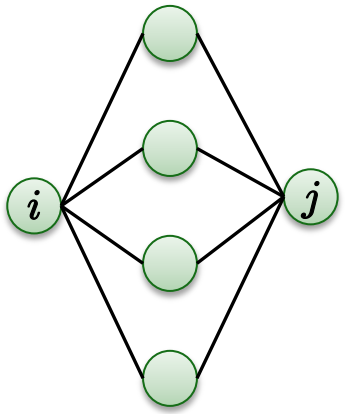
Sandia National Laboratories



Graph	Time (exact)	Path-sampling
Web-Berk	2 hrs	3 sec
Flickr	60 hrs	2 sec
Orkut	19 hrs	16 sec

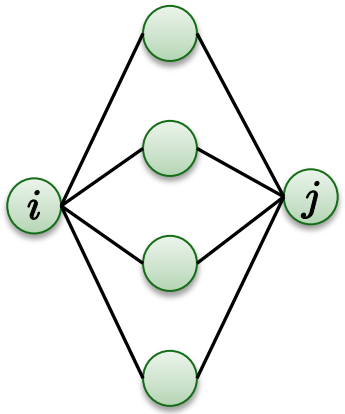
Jha, Seshadhri, Pinar, *WWW* '15

Finding Large Entries of A^2 (Binary)



$(A^2)_{ij}$ = number common nghbrs for vertex pair (i,j)

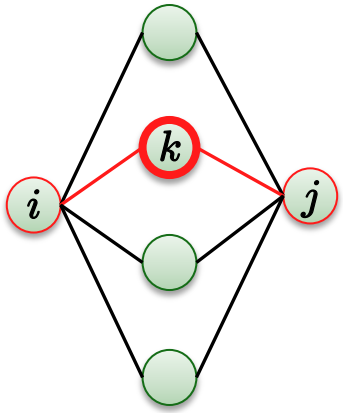
Finding Large Entries of A^2 (Binary)



$(A^2)_{ij}$ = number common nghbrs for vertex pair (i,j)

- Exact algorithm (C=A^2)
  - Sparse matrix-matrix multiply
    - Use Csparse (cs_multiply function) from Tim Davis
    - Computes one column of C at a time
    - May run out of memory
  - Modifications
    - Added user-defined threshold to remove small entries from each column as it is computed and therefore avoid out-of-memory errors
    - Exploit symmetry (only fill in upper portion of each column)

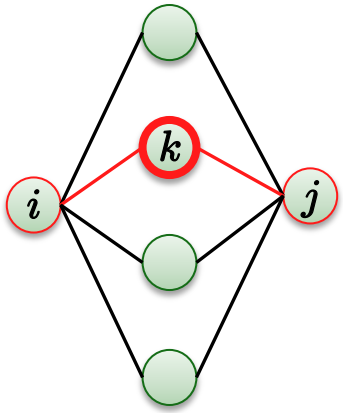
Wedge Sampling for Top-t Entries of $C=A^2$ (Binary)



Wedge Sampling for Top-t Entries of $C=A^2$ (Binary)

Algorithm Sample Wedges

1: $X \leftarrow 0$
2: **for** $\alpha = 1, \ldots, s$ **do**
3: Choose $k \propto d_k(d_k - 1)/2$
4: Choose $i \in \mathcal{N}(k)$
5: Choose $j \in \mathcal{N}(k) \setminus \{i\}$
6: $x_{ij} \leftarrow x_{ij} + 1$
7: **end for**
8: $X \leftarrow X + X^T$



- Basic Idea
  - s = # samples
  - t = # top dot products desired
  - $\beta\, t$ = budget for dot products
  - Choose wedges uniformly at random

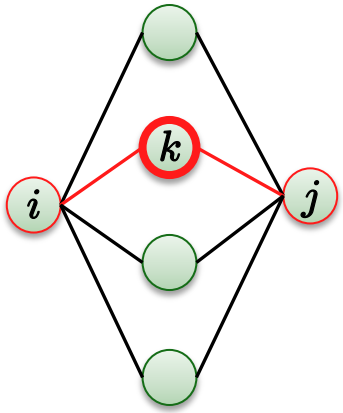
Wedge Sampling for Top-t Entries of $C=A^2$ (Binary)

Algorithm Sample Wedges

1: $X \leftarrow 0$
2: **for** $\alpha = 1, \ldots, s$ **do**
3: Choose $k \propto d_k(d_k - 1)/2$
4: Choose $i \in \mathcal{N}(k)$
5: Choose $j \in \mathcal{N}(k) \setminus \{i\}$
6: $x_{ij} \leftarrow x_{ij} + 1$
7: **end for**
8: $X \leftarrow X + X^T$

Algorithm Postprocess

1: $\mathcal{B} \leftarrow$ indices (i, j) corresponding to largest βt entries in $\texttt{triu}(X)$
2: Compute $\langle a_i, a_j \rangle \; \forall (i, j) \in \mathcal{B}$
3: Return top-t dot products and indices



$$\mathbb{E}(x_{ij}) \propto c_{ij} = \langle a_i, a_j \rangle$$

- Basic Idea
  - s = # samples
  - t = # top dot products desired
  - $\beta\, t$ = budget for dot products
  - Choose wedges uniformly at random

Wedge Sampling for Top-t Entries of $C=A^2$ (Binary)

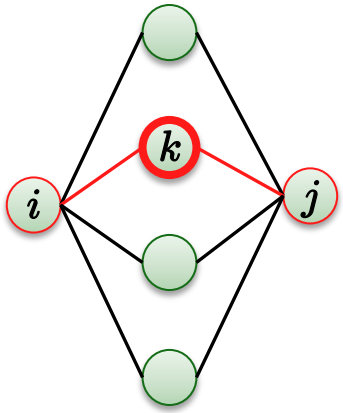
Sandia National Laboratories

Algorithm Sample Wedges

1: $X \leftarrow 0$
2: **for** $\alpha = 1, \ldots, s$ **do**
3: Choose $k \propto d_k(d_k - 1)/2$
4: Choose $i \in \mathcal{N}(k)$
5: Choose $j \in \mathcal{N}(k) \setminus \{i\}$
6: $x_{ij} \leftarrow x_{ij} + 1$
7: **end for**
8: $X \leftarrow X + X^T$

Algorithm Postprocess

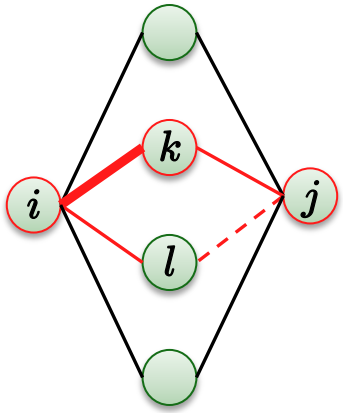
1: $\mathcal{B} \leftarrow$ indices (i, j) corresponding to largest βt entries in $\texttt{triu}(X)$
2: Compute $\langle a_i, a_j \rangle \; \forall (i, j) \in \mathcal{B}$
3: Return top-t dot products and indices



$$\mathbb{E}(x_{ij}) \propto c_{ij} = \langle a_i, a_j \rangle$$

- Basic Idea
  - s = # samples
  - t = # top dot products desired
  - $\beta\, t$ = budget for dot products
  - Choose wedges uniformly at random
- Cost
  - Preprocess: O($|E|$)
  - Sampling: O(s log(n))
  - Postprocess: O(s log(s) + $\beta\, tn$)

Diamond Sampling for Large Entries of A^2 (Binary)

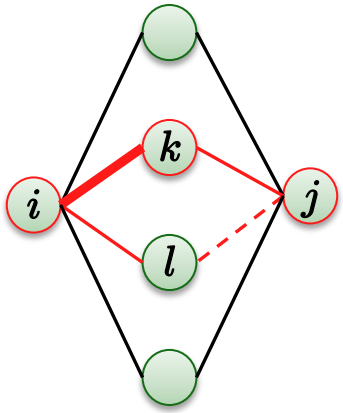


Diamond Sampling for Large Entries of A^2 (Binary)

Sandia National Laboratories

Algorithm Diamond Sampling

$X \leftarrow 0$
for $\alpha = 1, \ldots, s$ **do**
  Choose $(i,k) \in E$ $\propto (d_i - 1)(d_k - 1)$
  Choose $l \in \mathcal{N}(i) \setminus \{k\}$
  Choose $j \in \mathcal{N}(k) \setminus \{i\}$
  if $(l,j) \in E$ **then**
    $x_{ij} \leftarrow x_{ij} + 1$
    $x_{kl} \leftarrow x_{kl} + 1$
  end if
end for
$X \leftarrow X + X^T$



$$\mathbb{E}(x_{ij}) \propto c_{ij}^2 = (\langle a_i, a_j \rangle)^2$$

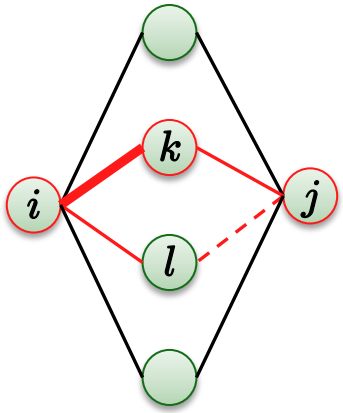
- Basic Idea
  - Choose three-path uniformly at random
  - Reject if not a 4-cycle
  - Result is uniform random 4-cycle

Diamond Sampling for Large Entries of A^2 (Binary)

Sandia National Laboratories

Algorithm Diamond Sampling

$X \leftarrow 0$
for $\alpha = 1, \ldots, s$ **do**
  Choose $(i,k) \in E$ $\propto (d_i - 1)(d_k - 1)$
  Choose $l \in \mathcal{N}(i) \setminus \{k\}$
  Choose $j \in \mathcal{N}(k) \setminus \{i\}$
  if $(l,j) \in E$ **then**
    $x_{ij} \leftarrow x_{ij} + 1$
    $x_{kl} \leftarrow x_{kl} + 1$
  end if
end for
$X \leftarrow X + X^T$



$$\mathbb{E}(x_{ij}) \propto c_{ij}^2 = (\langle a_i, a_j \rangle)^2$$

- Basic Idea
  - Choose three-path uniformly at random
  - Reject if not a 4-cycle
  - Result is uniform random 4-cycle
- Cost
  - Preprocess: O($|E|$)
  - Sampling: O(s log($|E|$))
  - Postprocess: O(s log(s) + $\beta\, tn$)

$C=A^2$ (Binary) Experiments

- s = # of samples
- t = # of top entries
- β = multiplier for dot-product budget
- $\mathcal{B}$ = set of most βt most frequent (i,j) pairs in sampling

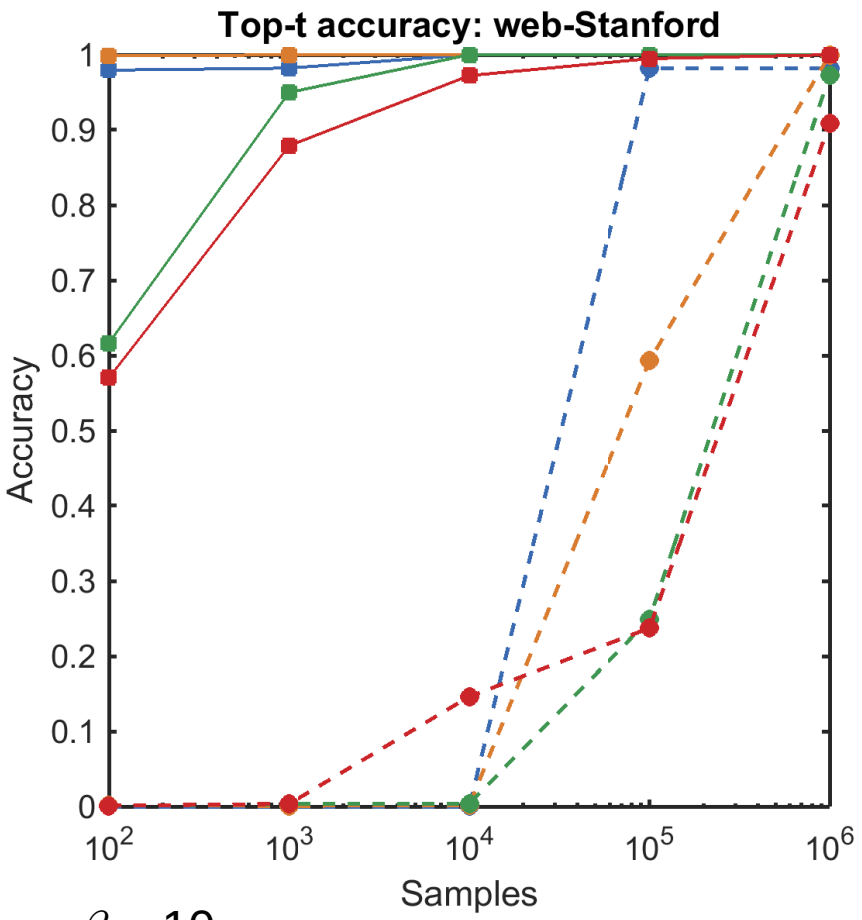
$$C = A^2$$

$$\hat{c}_{ij} = \begin{cases} c_{ij} & \text{if } (i,j) \in \mathcal{B} \\ 0 & \text{otherwise} \end{cases}$$

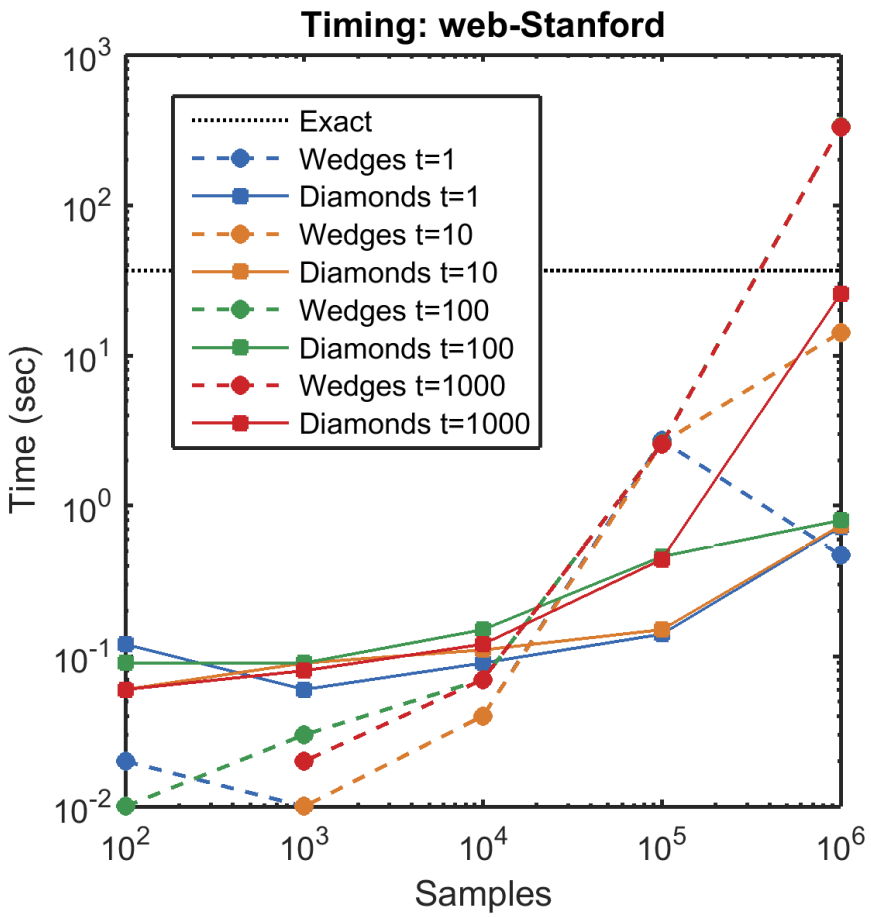
$$\text{score} = \frac{\max_{|\hat{\Omega}|=t} \left(\sum_{(i,j)\in\hat{\Omega}} \hat{c}_{ij}^2 \right)^{1/2}}{\max_{|\Omega|=t} \left(\sum_{(i,j)\in\Omega} c_{ij}^2 \right)^{1/2}}$$

Results: web-Stanford

Nodes: 2.82e+05 Edges: 1.99e+06



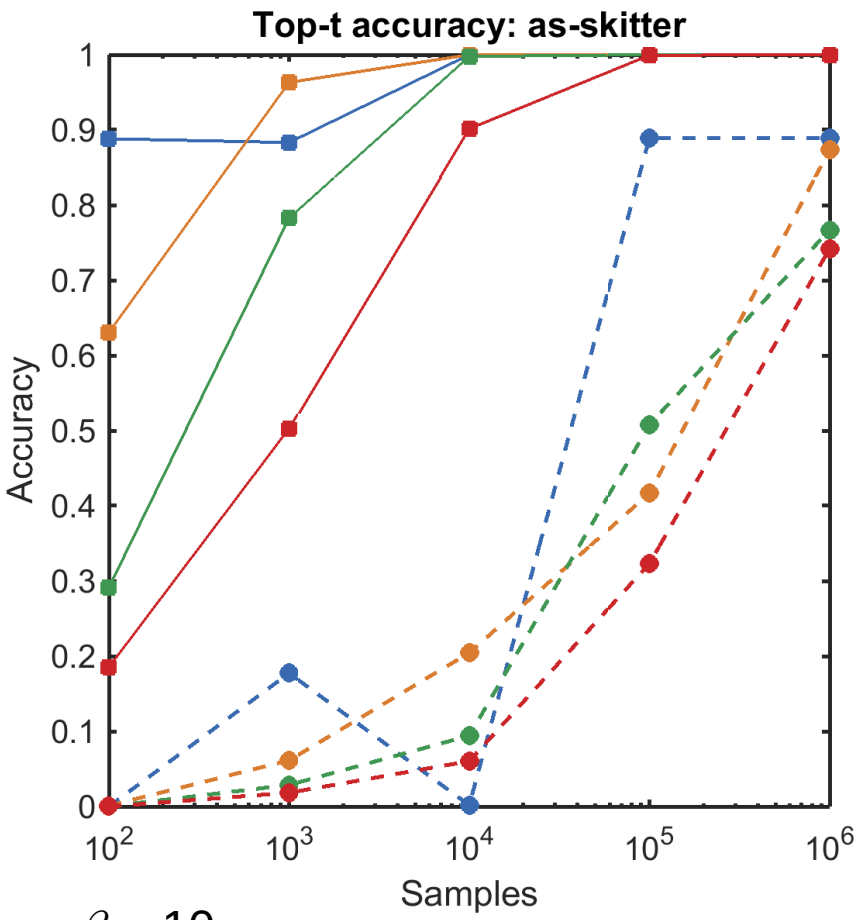
$\beta = 10$



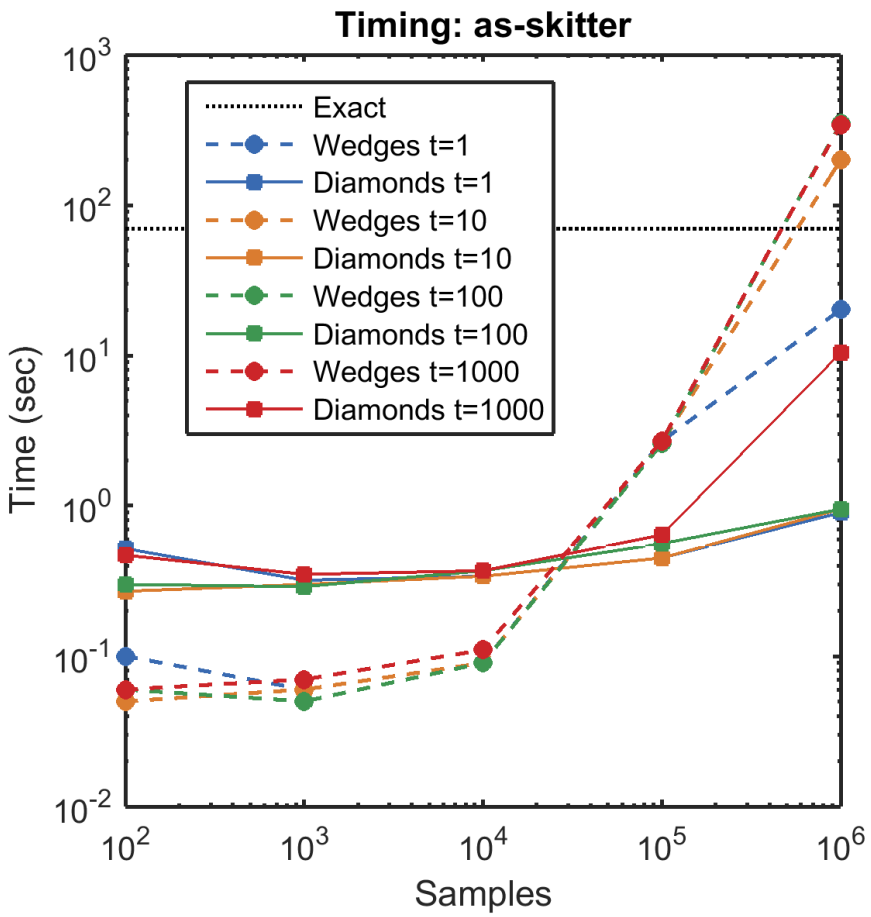
3-cycle closure rate: 30%

Results: as-skitter

Nodes: 1.70e+06 Edges: 2.22e+07 Wedges: 1.60e+10

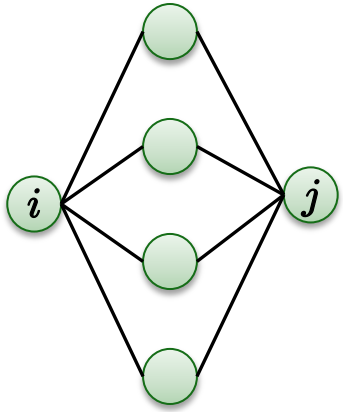


$\beta = 10$



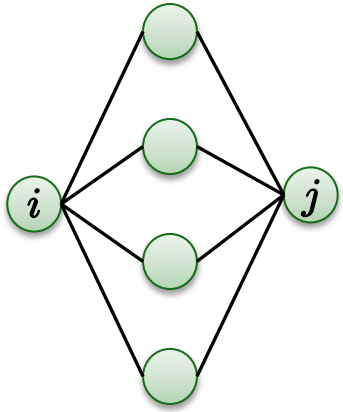
3-cycle closure rate: 17%

Finding Large Entries of A^2 (Weighted)



$$(A^2)_{ij} = \sum_k a_{ik} a_{kj}$$ = sum of all 2-path weight products for vertex pair (i,j)

Finding Large Entries of A^2 (Weighted)



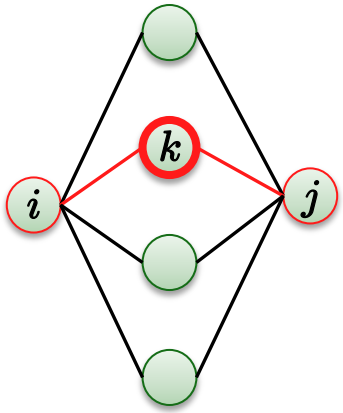
$$(A^2)_{ij} = \sum_k a_{ik} a_{kj}$$ = sum of all 2-path weight products for vertex pair (i,j)

$$\gamma = \sum_{i=1}^{n} \|a_i\|_1^2 = \sum_{ij} a_i^T a_j = \|A^2\|_1$$

Wedge Sampling for Top-t Entries of $C=A^2$ (Weighted)

Algorithm Sample Wedges

1: $X \leftarrow 0$
2: **for** $\alpha = 1, \ldots, s$ **do**
3: Choose $k \propto \|a_k\|_1^2/\gamma$
4: Choose $i \propto a_{ik}/\|a_k\|_1$
5: Choose $j \propto a_{jk}/\|a_k\|_1$
6: $x_{ij} \leftarrow x_{ij} + 1$
7: **end for**
8: $X \leftarrow X + X^T$



- Basic Idea
  - Weighted samples
  - Including diagonal entries of C
- Cost
  - Preprocess: O($|E|$)
  - Sampling: O(s log(n))
  - Postprocess: O(s log(s) + $\beta\, tn$)

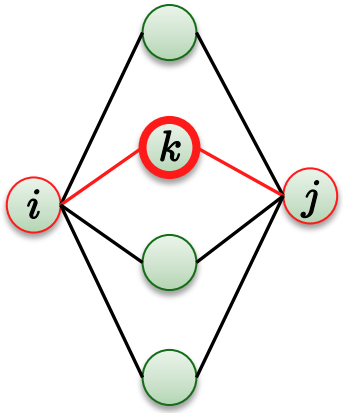
Wedge Sampling for Top-t Entries of $C=A^2$ (Weighted)

Sandia National Laboratories

Algorithm Sample Wedges

1: $X \leftarrow 0$
2: **for** $\alpha = 1, \ldots, s$ **do**
3: Choose $k \propto \|a_k\|_1^2/\gamma$
4: Choose $i \propto a_{ik}/\|a_k\|_1$
5: Choose $j \propto a_{jk}/\|a_k\|_1$
6: $x_{ij} \leftarrow x_{ij} + 1$
7: **end for**
8: $X \leftarrow X + X^T$

$$P(i,j) = \sum_k P(i,k,j)$$
$$= \sum_k \frac{\|a_k\|_1^2}{\gamma} \frac{a_{ik}}{\|a_k\|_1} \frac{a_{jk}}{\|a_k\|_1}$$
$$= a_i^T a_j/\gamma$$



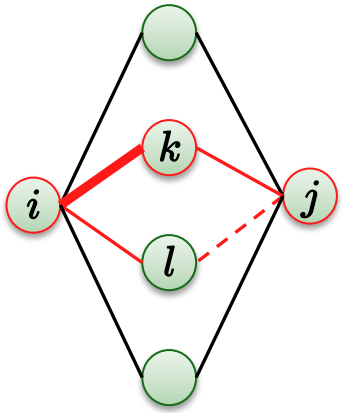
$$\mathbb{E}(x_{ij}) \propto c_{ij} = \langle a_i, a_j \rangle$$

- Basic Idea
  - Weighted samples
  - Including diagonal entries of C
- Cost
  - Preprocess: O($|E|$)
  - Sampling: O(s log(n))
  - Postprocess: O(s log(s) + $\beta\, tn$)

Diamond Sampling for Large Entries of A^2 (Weighted)

Algorithm Diamond Sampling

$X \leftarrow 0$
for $(i,k) \in E$ **do**
  $w_{ik} \leftarrow a_{ik}\|a_i\|_1\|a_k\|_1$
end for
$\omega \leftarrow \sum_{ik} w_{ik}$
for $\alpha = 1, \ldots, s$ **do**
  Choose $(i,k) \propto w_{ik}/\omega$
  Choose $l \propto a_{il}/\|a_i\|_1$
  Choose $j \propto a_{kj}/\|a_k\|_1$
  $x_{ij} \leftarrow x_{ij} + a_{jl}$
  $x_{kl} \leftarrow x_{kl} + a_{jl}$
end for
$X \leftarrow X + X^T$



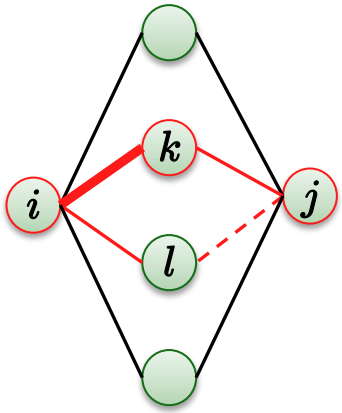
- Cost
  - Preprocess: O($|E|$)
  - Sampling: O(s log($|E|$) + s log(n))
  - Postprocess: O(s log(s) + $\beta\, tn$)

Note: Can also be adapted for negative entries, but omitting here due to space.

Diamond Sampling for Large Entries of A^2 (Weighted)

Algorithm Diamond Sampling

$X \leftarrow 0$
for $(i,k) \in E$ **do**
  $w_{ik} \leftarrow a_{ik}\|a_i\|_1\|a_k\|_1$
end for
$\omega \leftarrow \sum_{ik} w_{ik}$
for $\alpha = 1, \ldots, s$ **do**
  Choose $(i,k) \propto w_{ik}/\omega$
  Choose $l \propto a_{il}/\|a_i\|_1$
  Choose $j \propto a_{kj}/\|a_k\|_1$
  $x_{ij} \leftarrow x_{ij} + a_{jl}$
  $x_{kl} \leftarrow x_{kl} + a_{jl}$
end for
$X \leftarrow X + X^T$



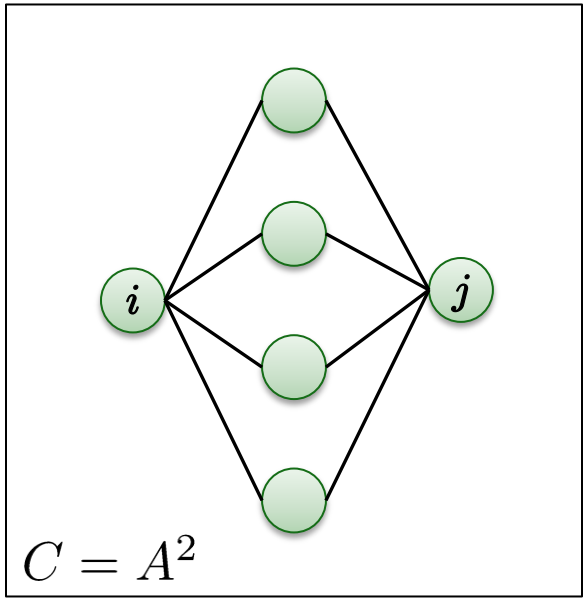
$$\mathbb{E}(x_{kl}) = \sum_{ij} P(l-i-k-j)a_{jl}$$
$$= \sum_{ij} \frac{a_{ik}\|a_i\|_1\|a_k\|_1}{\omega} \frac{a_{il}}{\|a_i\|_1} \frac{a_{kj}}{\|a_k\|_1} a_{jl}$$
$$= \frac{1}{\omega} \sum_{ij} a_{ik}a_{il}a_{jk}a_{jl}$$
$$= \frac{1}{\omega} \sum_{i} a_{ik}a_{il} \sum_{j} a_{jk}a_{jl}$$
$$= \frac{(a_k^T a_l)^2}{\omega} \propto c_{kl}^2$$

- Cost
  - Preprocess: O($|E|$)
  - Sampling: O(s log($|E|$) + s log(n))
  - Postprocess: O(s log(s) + $\beta\, tn$)

Note: Can also be adapted for negative entries, but omitting here due to space.

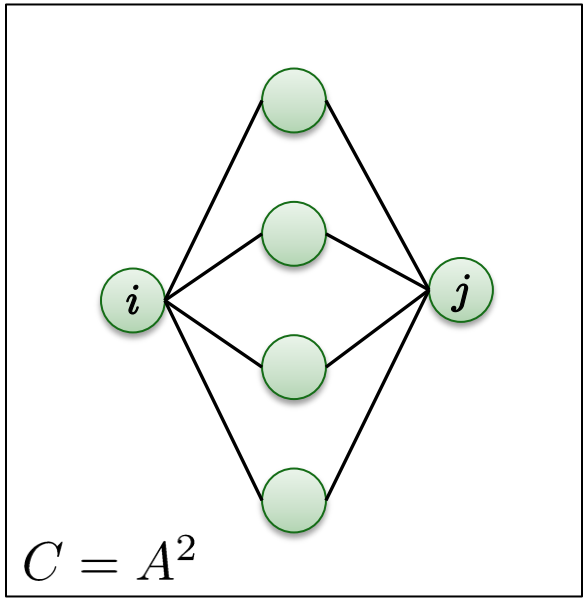
Extensions: A^TA and A^TB

Case 1: Undirected Graph

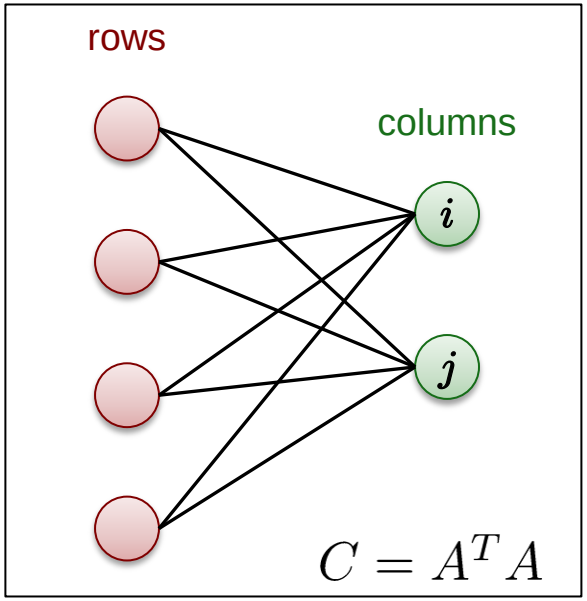


Extensions: A^TA and A^TB

Case 1: Undirected Graph

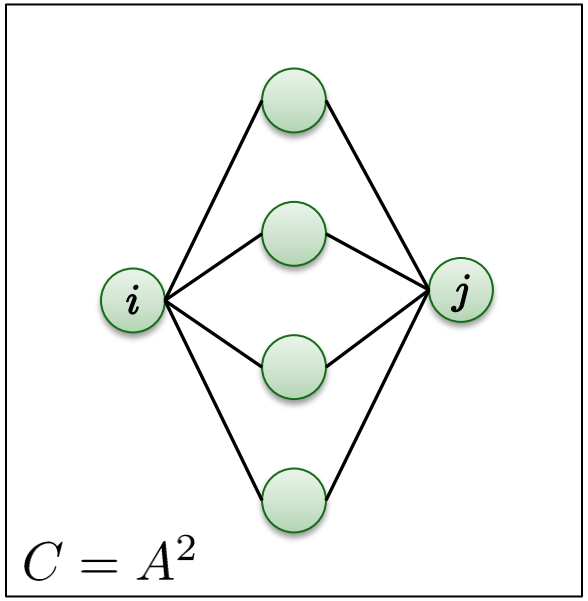


Case 2: Bipartite Graph

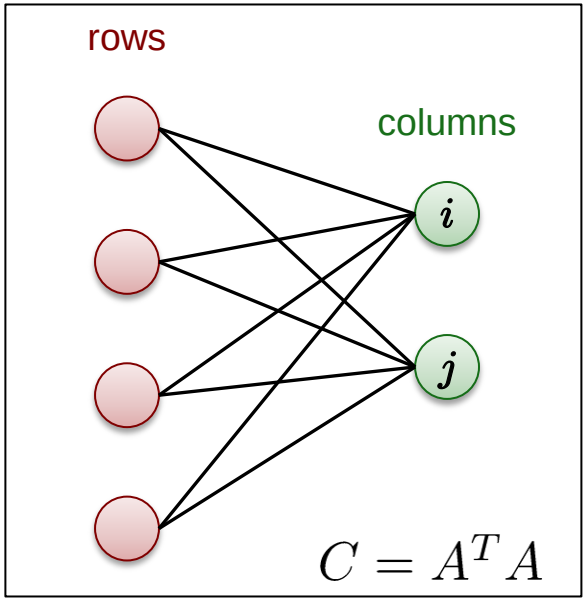


Extensions: A^TA and A^TB

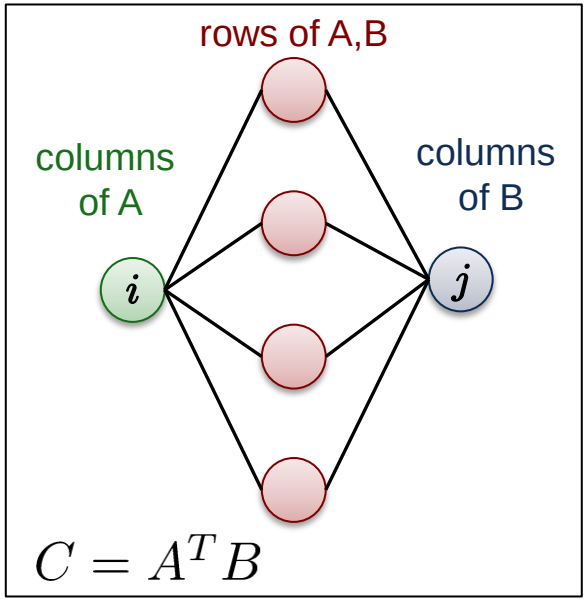
Case 1: Undirected Graph



Case 2: Bipartite Graph



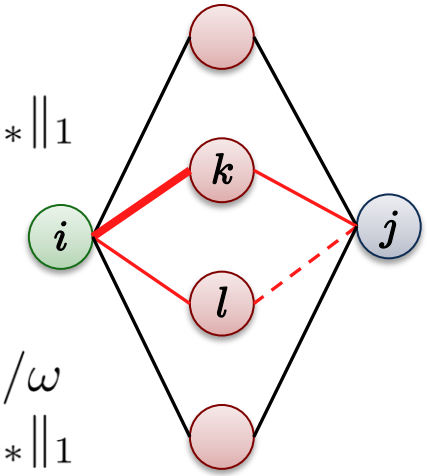
Case 3: Tripartite Graph



Diamond Sampling for Large Entries of A^TB (Weighted)

Algorithm Diamond Sampling

$X \leftarrow 0$
for $(i,k) \in E_A$ **do**
  $w_{ik} \leftarrow a_{ki}\|a_{*i}\|_1\|b_{k*}\|_1$
end for
$\omega \leftarrow \sum_{ik} w_{ik}$
for $\alpha = 1, \ldots, s$ **do**
  Choose $(i,k) \propto w_{ik}/\omega$
  Choose $j \propto b_{kj}/\|b_{k*}\|_1$
  Choose $l \propto a_{li}/\|a_{*i}\|_1$
  $x_{ij} \leftarrow x_{ij} + b_{lj}$
end for



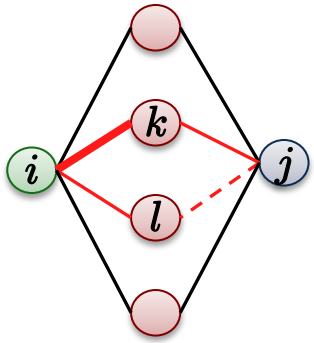
$$\mathbb{E}(x_{ij}) = \sum_{kl} P(l-i-k-j)b_{lj}$$
$$= \sum_{kl} \frac{a_{ki}\|a_{*i}\|_1\|b_{k*}\|_1}{\omega} \frac{b_{kj}}{\|b_{k*}\|_1} \frac{a_{li}}{\|a_{*i}\|_1} b_{lj}$$
$$= \frac{1}{\omega} \sum_{kl} a_{ki}b_{kj}a_{li}b_{lj}$$
$$= \frac{1}{\omega} \sum_{k} a_{ki}b_{kj} \sum_{l} a_{li}b_{lj}$$
$$= \frac{(a_i^T b_j)^2}{\omega} \propto c_{ij}^2$$

- Cost
  - Preprocess: O($|E|$)
  - Sampling: O(s log($|E|$) + s log(n))
  - Postprocess: O(s log(s) + $\beta\, tn$)

Sampling on Graphs is a Powerful Tool for Big Data

Sandia National Laboratories

- Sampling methods have proved useful for…
  - Triangle Counting
  - Four-vertex Patterns
- Currently developing sampling methods for calculating largest entries
  - $C = A^2$, $C = A^T B$
  - Determine number of samples
  - Bound error
  - Efficient implementation
  - Seeking more applications

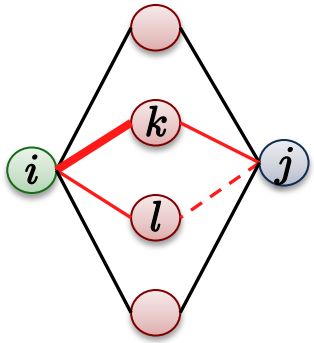


- Co-authors
  - Grey Ballard
  - Ali Pinar
  - C. “Sesh” Seshadhri
- Acknowledgments
  - Madhav Jha
  - Kevin Matulef
  - Jon Berry
  - Cindy Phillips
  - Todd Plantenga

Sampling on Graphs is a Powerful Tool for Big Data

Sandia National Laboratories

- Sampling methods have proved useful for…
  - Triangle Counting
  - Four-vertex Patterns
- Currently developing sampling methods for calculating largest entries
  - $C = A^2$, $C = A^T B$
  - Determine number of samples
  - Bound error
  - Efficient implementation
  - Seeking more applications



- Co-authors
  - Grey Ballard
  - Ali Pinar
  - C. "Sesh" Seshadhri
- Acknowledgments
  - Madhav Jha
  - Kevin Matulef
  - Jon Berry
  - Cindy Phillips
  - Todd Plantenga

Thank you!

Alias Method

- Reference: Vose, Michael D., A Linear Algorithm For Generating Random Numbers With a Given Distribution. *IEEE Transactions on Software Engineering*, **17**(9) (September 1991):972-975.
- https://possiblywrong.wordpress.com/2012/02/05/the-alias-method-and-double-precision/

Back-up slide: Number 4-cycles

F. Harary and B. Manvel, ***On the number of cycles in a graph***, Matematický casopis 21(1):1971, http://dml.cz/dmlcz/126802

Theorem 1. *If G is a (p, q) graph with adjacency matrix A, then the number of 4-cycles in G is:*

$$c_4(G) = \tfrac{1}{8}\,[\mathrm{tr}(A^4) - 2q - 2\sum_{i\neq j} a_{ij}^{(2)}]. \tag{2}$$

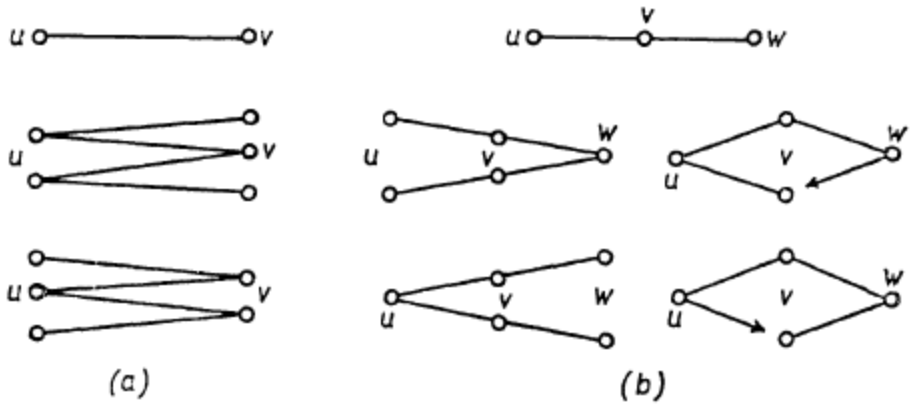


Fig. 1. Shapes of non-cyclic closed 4-walks in a graph.