Smooth Kronecker

SYSTOPIA

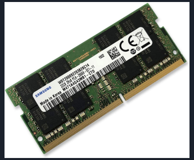
- Vaastav Anand
- Puneet Mehrotra
- Daniel Margo
- Margo Seltzer

THE UNIVERSITY OF BRITISH COLUMBIA

Graph Systems have taken over the world

- Graph Data is everywhere
- Exponential growth in research of distributed graph processing systems since Pregel (2010)
- Graph Processing Systems must scale to very large graphs

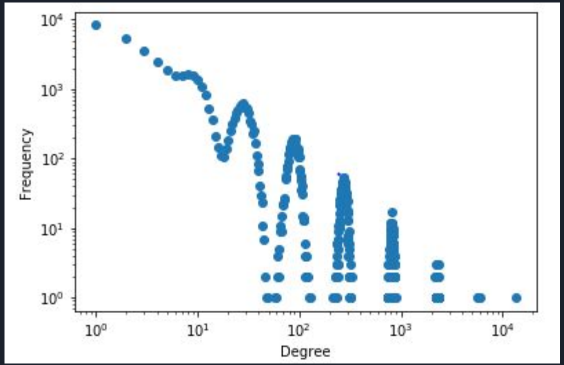
Widely used graphs are small

Dataset	# Vertices	# Edges	Size	Device whose memory dataset will fit in
cit-Patents	~3.7M	~16.5M	289MB	iPhone 4 
soc-LiveJournal	~4.8M	~68.9M	1.1GB	iPhone 7 
Twitter-2010	~41M	~1.4B	26GB	Our advisor's laptop 

Kronecker Generators save the day

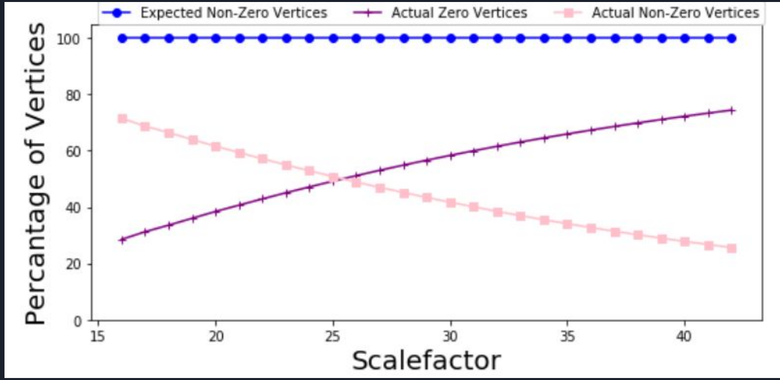
- R-Mat and Kronecker Graph generators have been popularly used for generating large graphs.
- They are easy to use and highly parallelizable which makes them perfect as the data source for scalability experiments.
- Edges in the graph are generated according to a 2 x 2 seed distribution where each element in the distribution approximately corresponds to the fraction of edges in a particular quadrant of the adjacency matrix of the graph.

Do they really save the day?

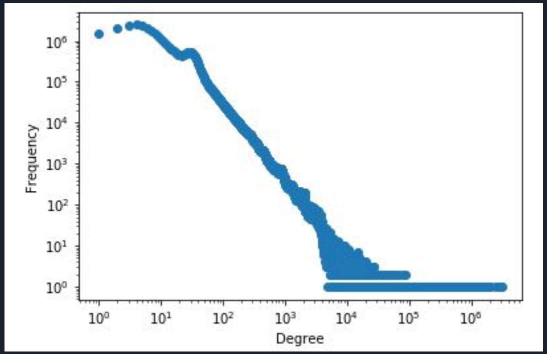


Kronecker Graph generators suffer from 2 major problems that make them inappropriate for scalability benchmarking of graph processing systems

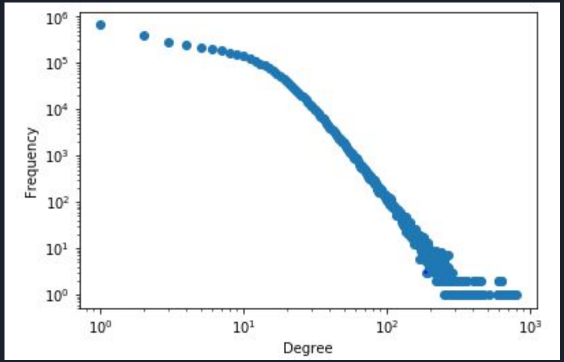
- The degree distributions of Kronecker graphs are combed, unlike any real graph
- As scale increases, so does the fraction of 0-degree vertices.
- **Implication**: a scale of 30 produces a graph with only 400M non-0 degree vertices, not the expected 1 Billion



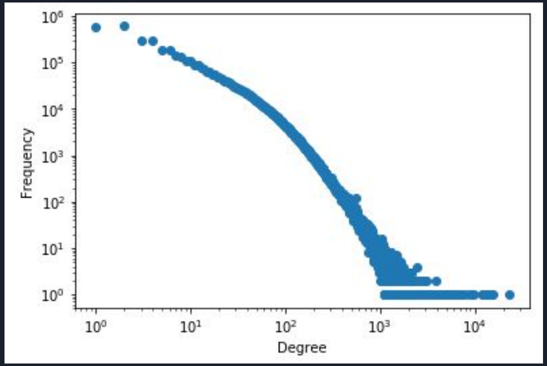
Which one of these is not like the others?



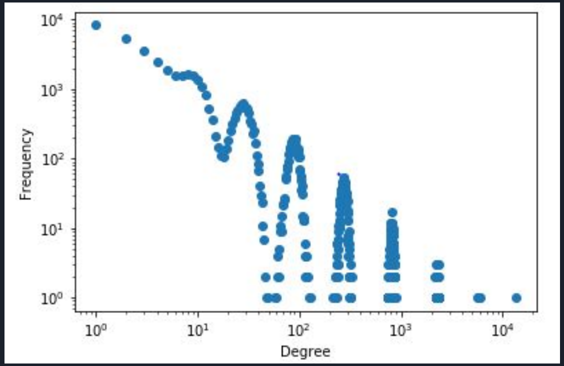
Twitter-2010



Cit-Patents

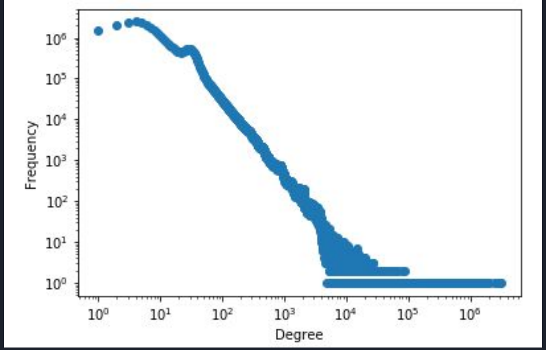


Soc-Livejournal

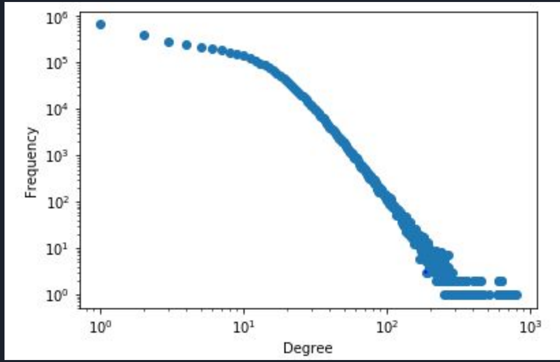


Kronecker (Snap)

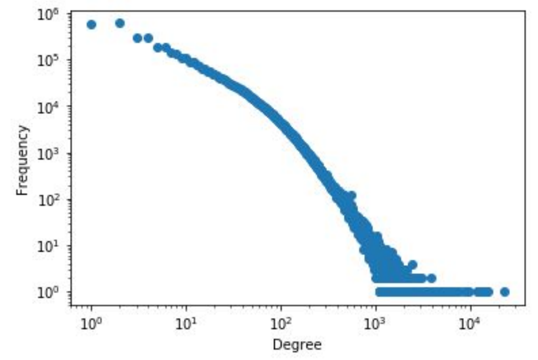
Which one of these is not like the others?



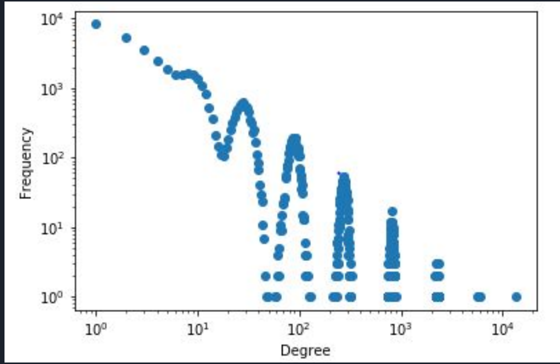
Twitter-2010



Cit-Patents

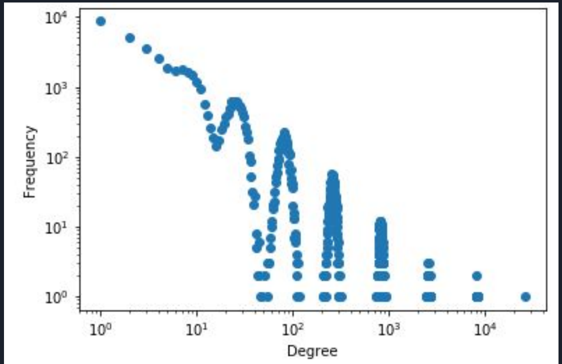


Soc-Livejournal

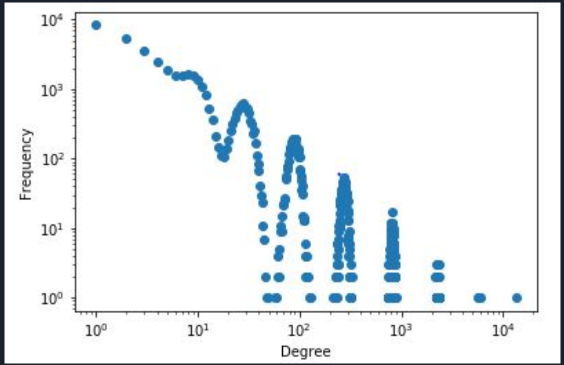


Kronecker (Snap)

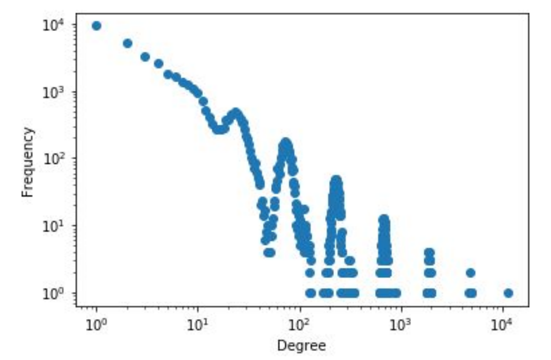
Degree distribution combing is pervasive



Graph500



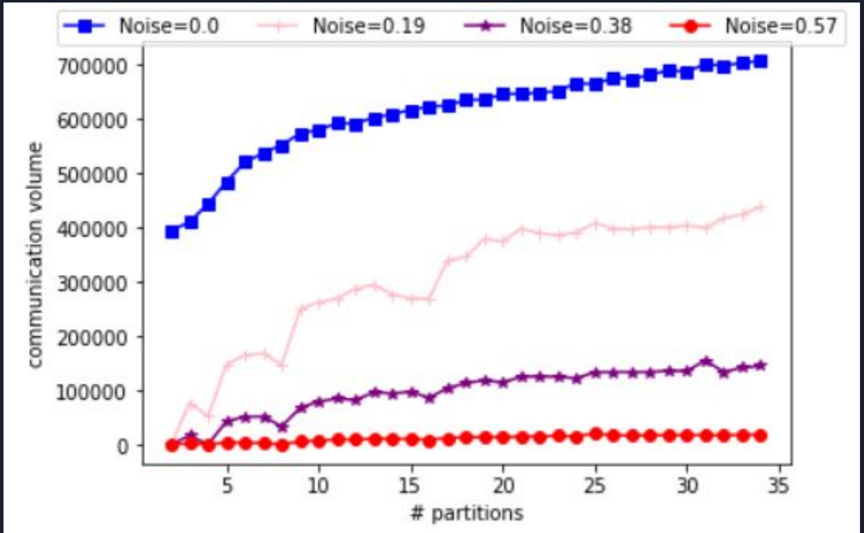
Snap-Kron



TrillionG

Adding Noise

- Seshadhri et al. proposed smoothing by blurring kronecker iterations with uniform random noise.
- It is not completely obvious how much noise to add
- Adding too much noise can be catastrophic as it can drastically affect the partitionability of the graph.



Smooth Kronecker Generator

Problems with using the Kronecker-generated graphs for benchmarking

- The Kronecker model follows a reasonable degree distribution; however it is undersampled as each vertex's degree is determined by the non-unique sum of its binary representation.
- For Noisy Kronecker, the vertex degrees vary normally around the undersampled model, instead of sampling the correct underlying model.

Solution:

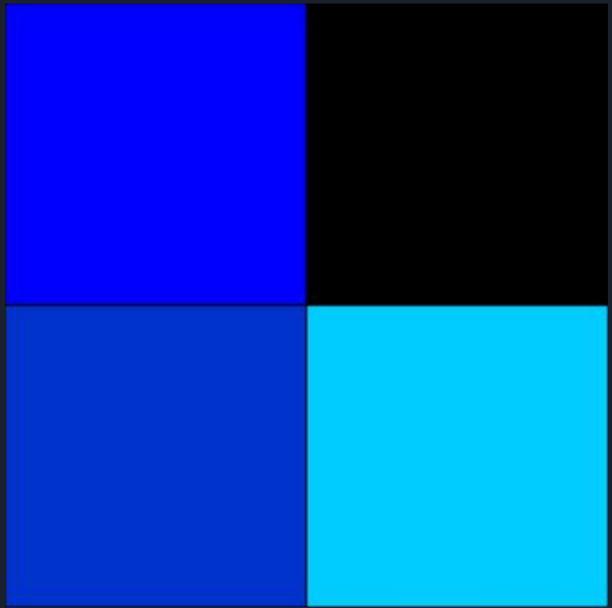
- Resample a different small distribution from the correct underlying model and mix it with the original 2x2 seed to smoothen the undersampling.

The Kronecker Product

- Mathematically, it is a product between two matrices where each element in a matrix is multiplied with the whole other matrix.

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 1\begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} & 2\begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} \\ 3\begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} & 4\begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1\times 0 & 1\times 5 & 2\times 0 & 2\times 5 \\ 1\times 6 & 1\times 7 & 2\times 6 & 2\times 7 \\ 3\times 0 & 3\times 5 & 4\times 0 & 4\times 5 \\ 3\times 6 & 3\times 7 & 4\times 6 & 4\times 7 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{bmatrix}.$$



Kronecker multiplication in 1-Dimension

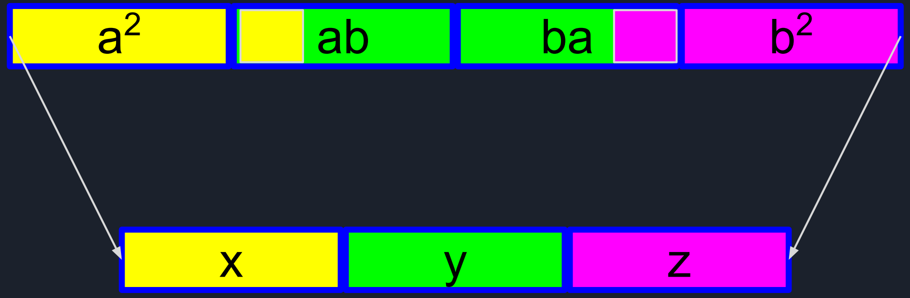
a	b

a^2	ab	ba	b^2

a^3	a^2b	a^2b	ab^2	ba^2	b^2a	b^2a	b^3

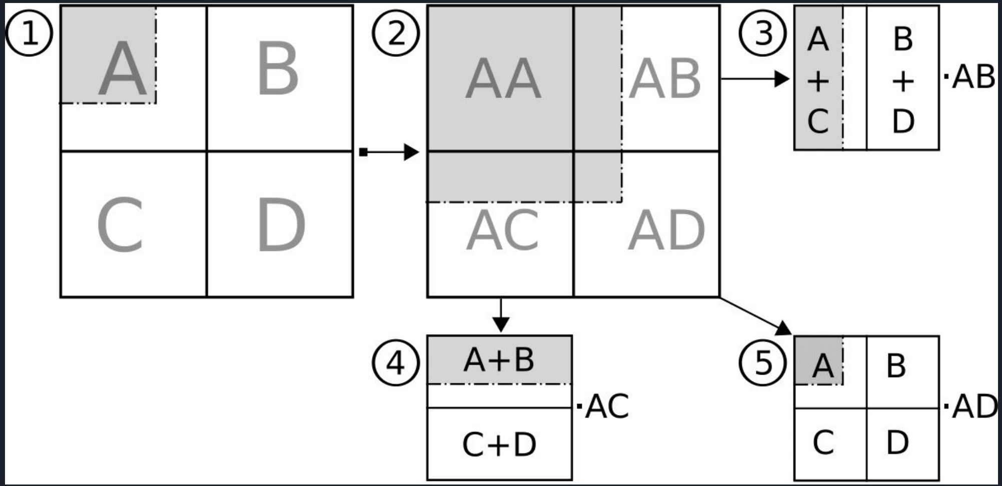
Resampling in 1-Dimension

- To resample the initial 1x2 seed [a,b] as a 1x3 seed [x,y,z], we sample the leftmost third (x), the middle third (y), and the rightmost third (z) as the Kronecker product reaches infinity.
- For the leftmost third, i.e. x, the value is just the geometric series with initial term **aa** and ratio **ab.**
- For the rightmost third, i.e. z, the value is just the geometric series with initial term **bb** and ratio **ab**
- y is simply calculated as 1-x-z.



2x2 Seed Resampling

- We now need to resample a 2x2 seed as a 3x3 seed.
- We follow the same idea from the 1D sampling where we sample each element of the 3x3 seed as the Kronecker product of the 2x2 seed goes to infinity.
- For eg: The first element is sampled as the leftmost corner goes to infinity.



Smooth Kronecker Algorithm

1. Given a 2x2 seed, number of edges, scalefactor2 and scalefactor3, first resample a 3x3 seed.

- Scalefactor2 and scalefactor3 collectively decide the number of vertices in the matrix.
- The total number of vertices is given by $2^{\text{scalefactor2}} * 3^{\text{scalefactor3}}$.
- Generate an edge using scalefactor2 + scalefactor3 Kronecker iterations.

a	b
c	d

2 x 2 Seed

r	s	t
u	v	w
x	y	z

3 x 3 Seed

Smooth Kronecker Algorithm

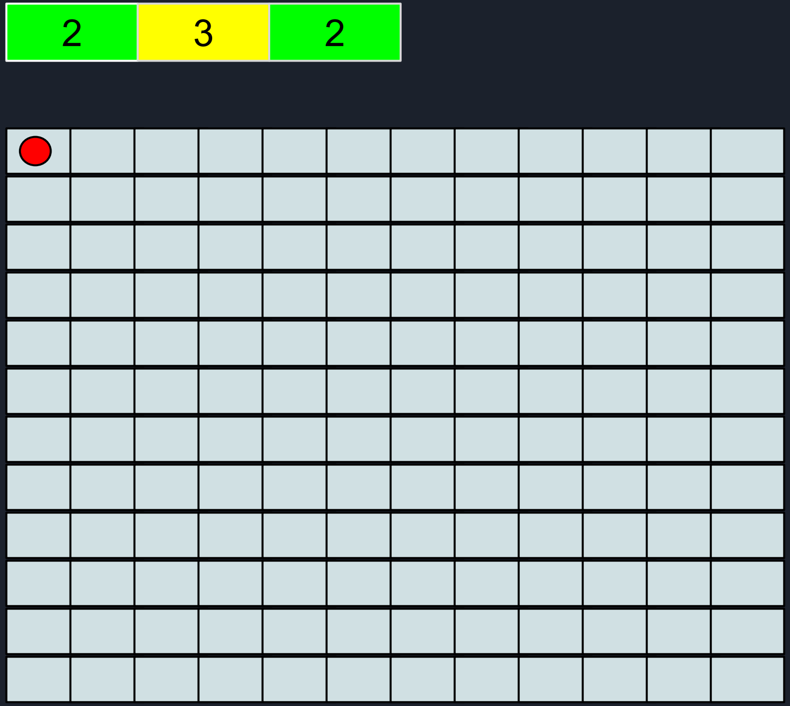
1. Given a 2x2 seed, number of edges, scalefactor2 and scalefactor3, first resample a 3x3 seed.

2. For each edge: randomly determine which seed will be used for sampling at every Kronecker iteration.

- E.g., For scalefactor2=4 and scalefactor3=2, one possible order is shown on the right.

2	Iteration #1
2	Iteration #2
2	Iteration #3
3	Iteration #4
2	Iteration #5
3	Iteration #6

Smooth Kronecker Algorithm



1. Given a 2x2 seed, number of edges, scalefactor2 and scalefactor3, first resample a 3x3 seed.

2. For each edge: randomly determine which seed will be used for sampling at every Kronecker iteration.

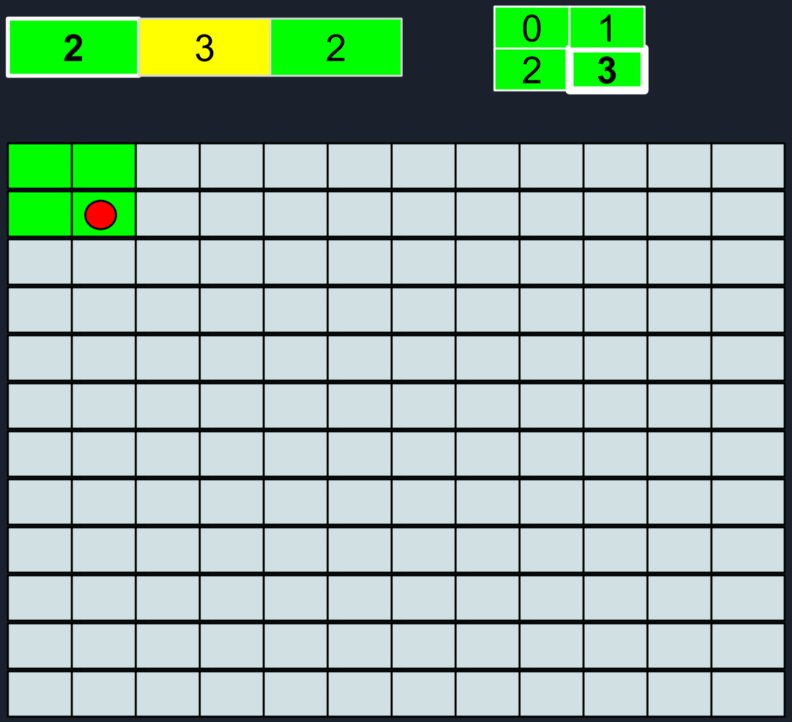
3. Recursively sample from the seed according to the operation order to obtain the source and destination of the edge.

Smooth Kronecker Algorithm

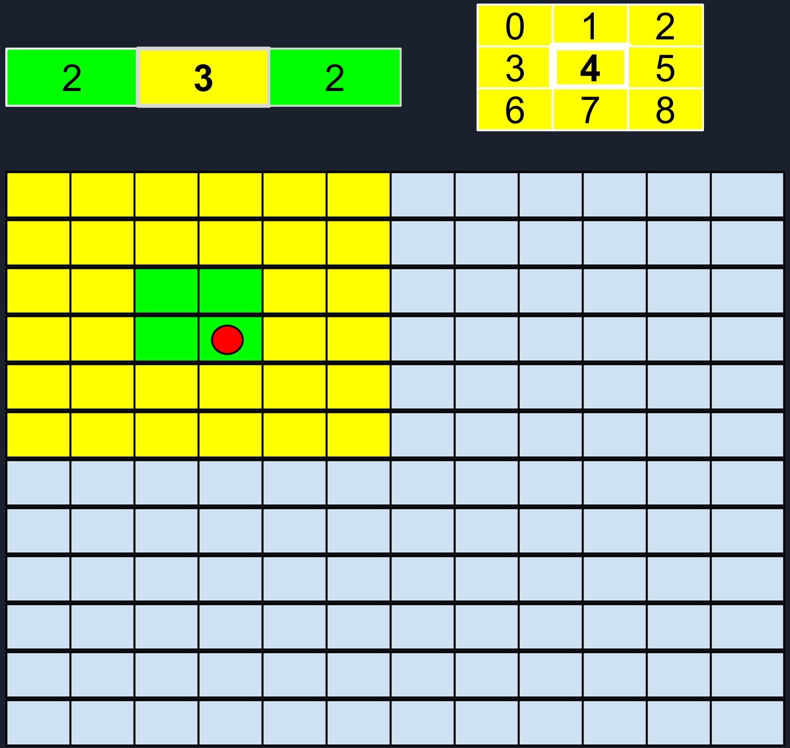
1. Given a 2x2 seed, number of edges, scalefactor2 and scalefactor3, first resample a 3x3 seed.

2. For each edge: randomly determine which seed will be used for sampling at every Kronecker iteration.

3. Recursively sample from the seed according to the operation order to obtain the source and destination of the edge.



Smooth Kronecker Algorithm



1. Given a 2x2 seed, number of edges, scalefactor2 and scalefactor3, first resample a 3x3 seed.

2. For each edge: randomly determine which seed will be used for sampling at every Kronecker iteration.

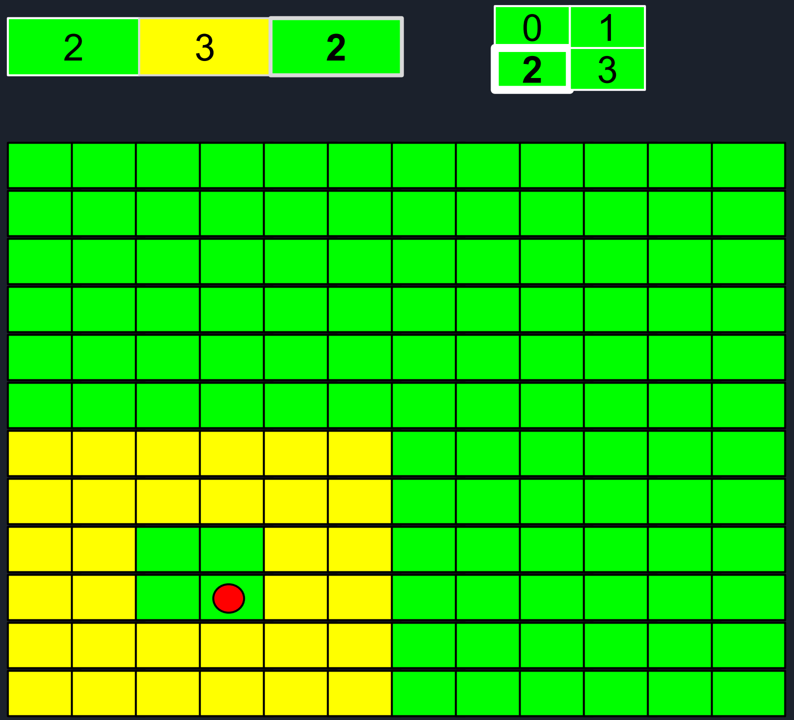
3. Recursively sample from the seed according to the operation order to obtain the source and destination of the edge.

Smooth Kronecker Algorithm

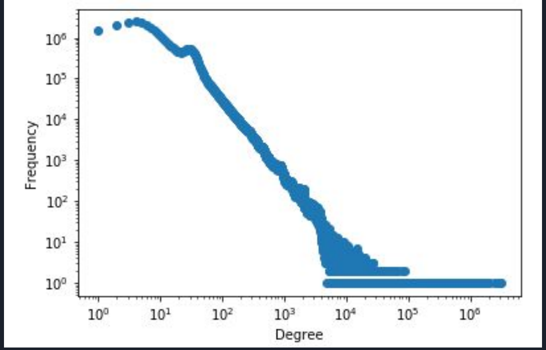
1. Given a 2x2 seed, number of edges, scalefactor2 and scalefactor3, first resample a 3x3 seed.

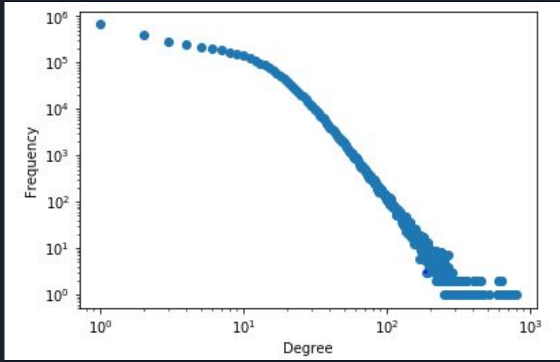
2. For each edge: randomly determine which seed will be used for sampling at every Kronecker iteration.

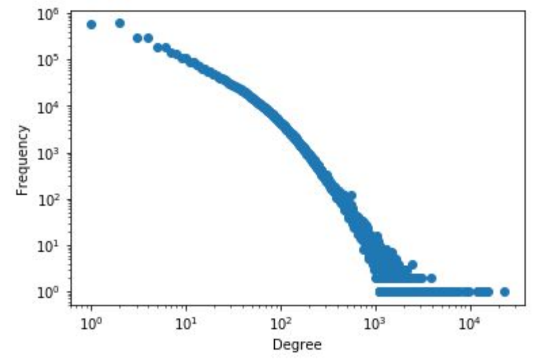
3. Recursively sample from the seed according to the operation order to obtain the source and destination of the edge.

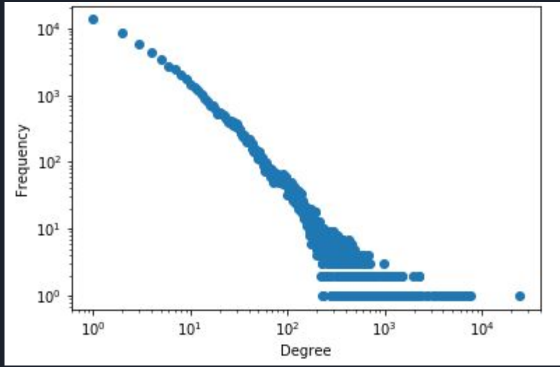


Smooth Kronecker looks like real graphs

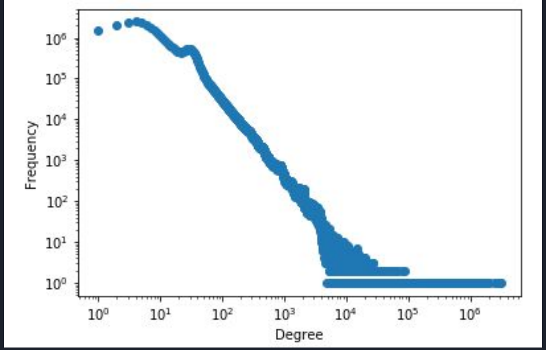




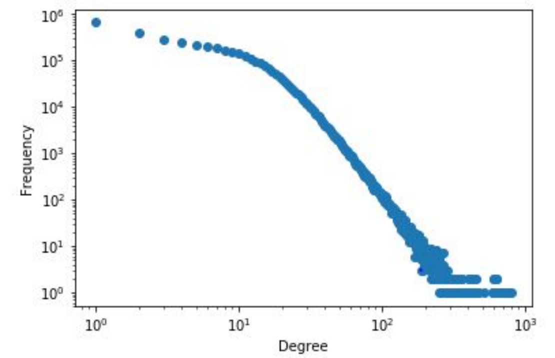




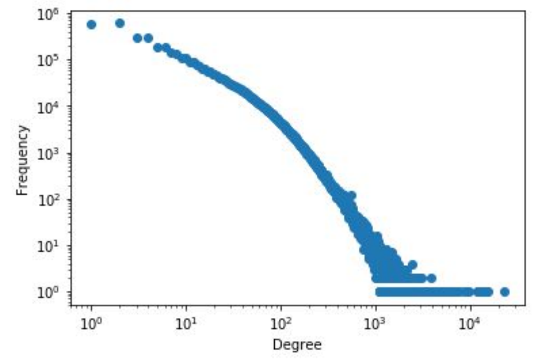
Smooth Kronecker looks like real graphs



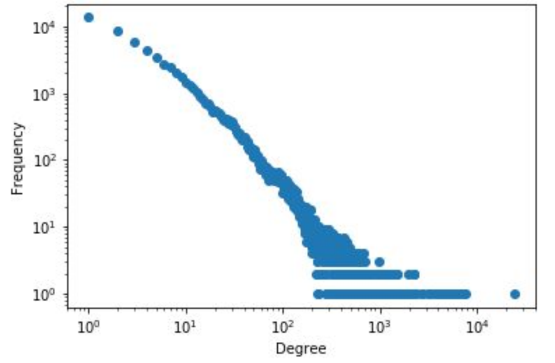
Twitter-2010



Cit-Patents



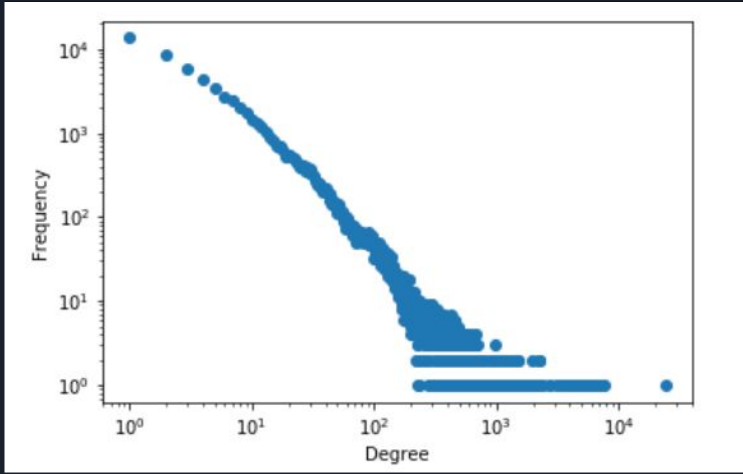
Soc-Livejournal



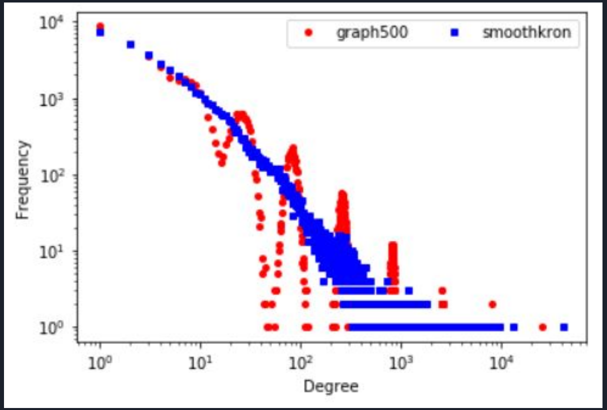
Smooth-Kron

Large Variety of Scales

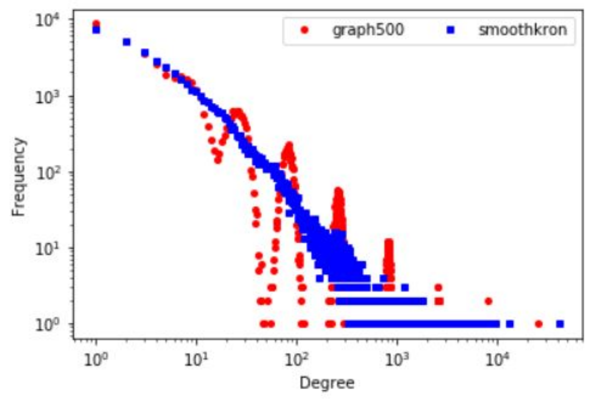
- With Smooth Kronecker, we are no longer restricted to powers of 2. Current implementation supports number of vertices of the form $2^x * 3^y$.
- To achieve smoothening of the degree distribution, only 1 resampled seed is required.
- Smooth Kronecker provides finer control over the number of vertices for potential scalability experiments
- With other resamplings, other scales can also be supported.

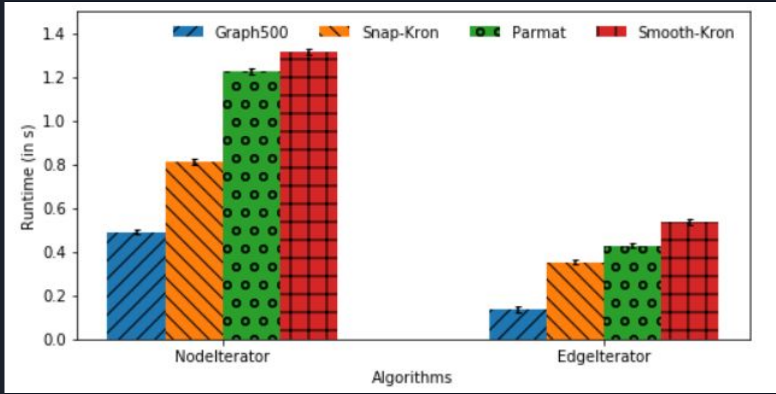


Smooth Kron to rule them all

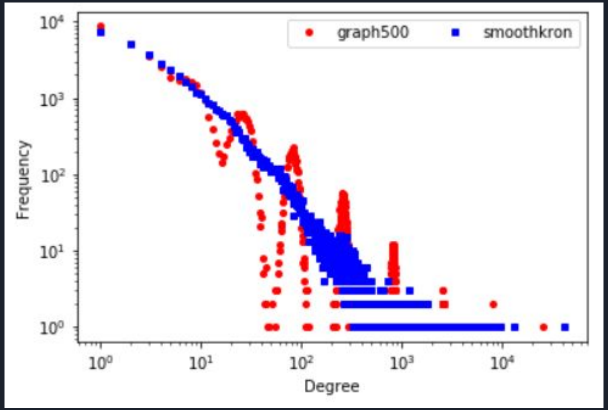


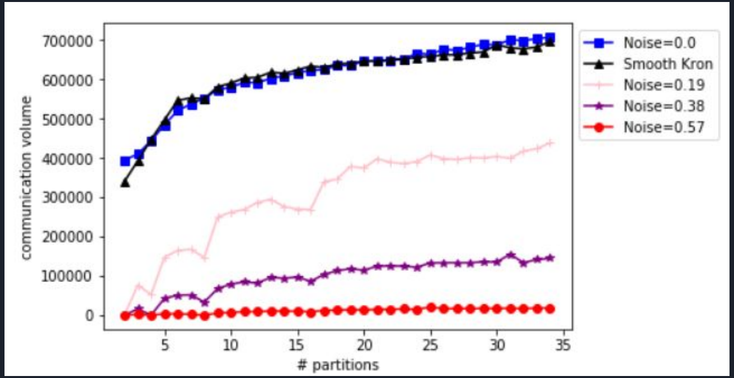
Smooth Kron to rule them all

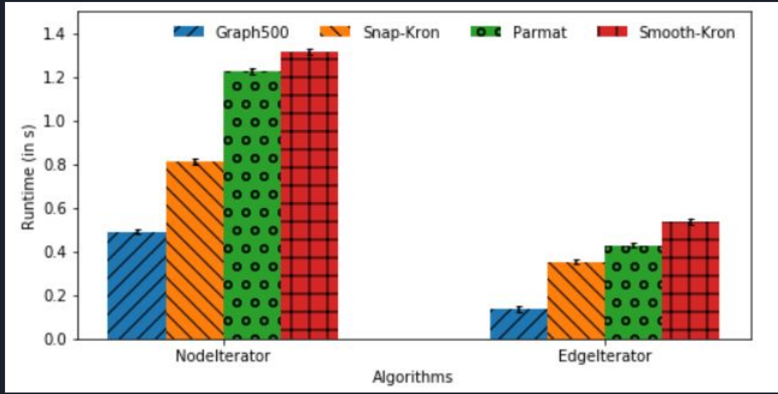




Smooth Kron to rule them all

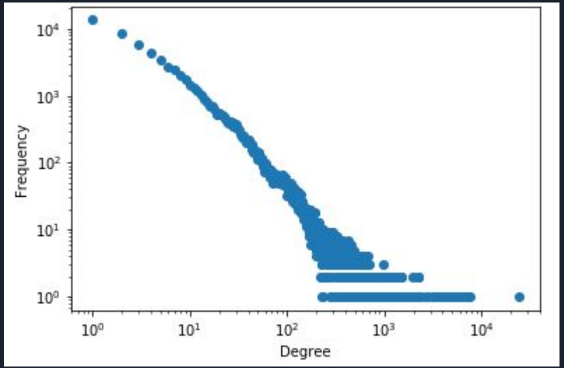






Conclusion

- We strongly urge the community to use the **Smooth Kronecker Generator** for generating synthetic graphs.
- **Smooth Kronecker Generator** is open source and available at https://github.com/dmargo/smooth_kron_gen



"Decrease Noise Pollution, Use Smooth Kronecker!"