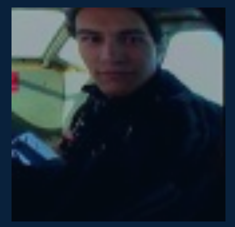
Darwini: Generating realistic large-scale social graphs

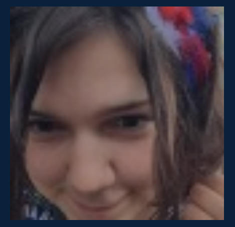
Sergey Edunov
Facebook



Dionysios Logothetis
Facebook

Cheng Wang
University of Houston

Avery Ching
Facebook



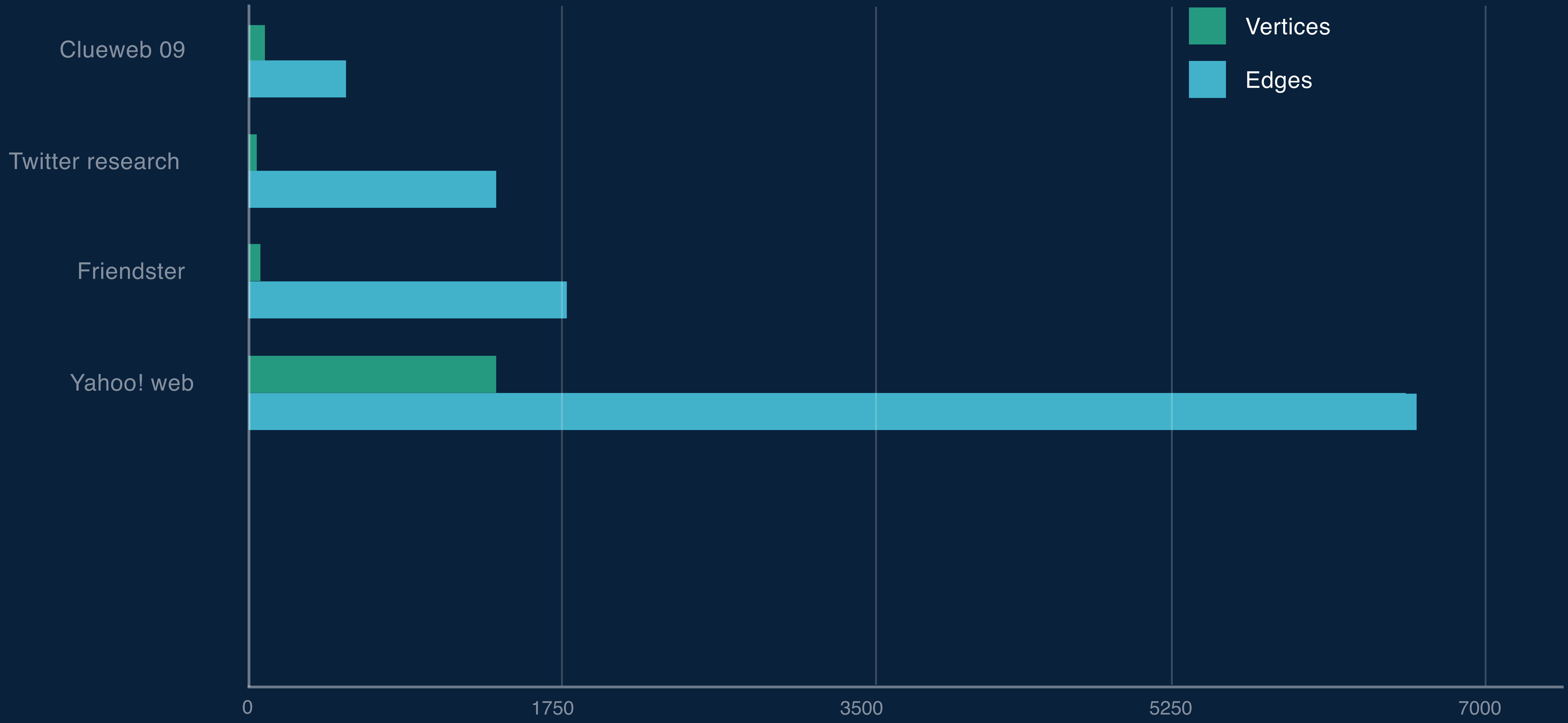
Maja Kabiljo
Facebook

Why?

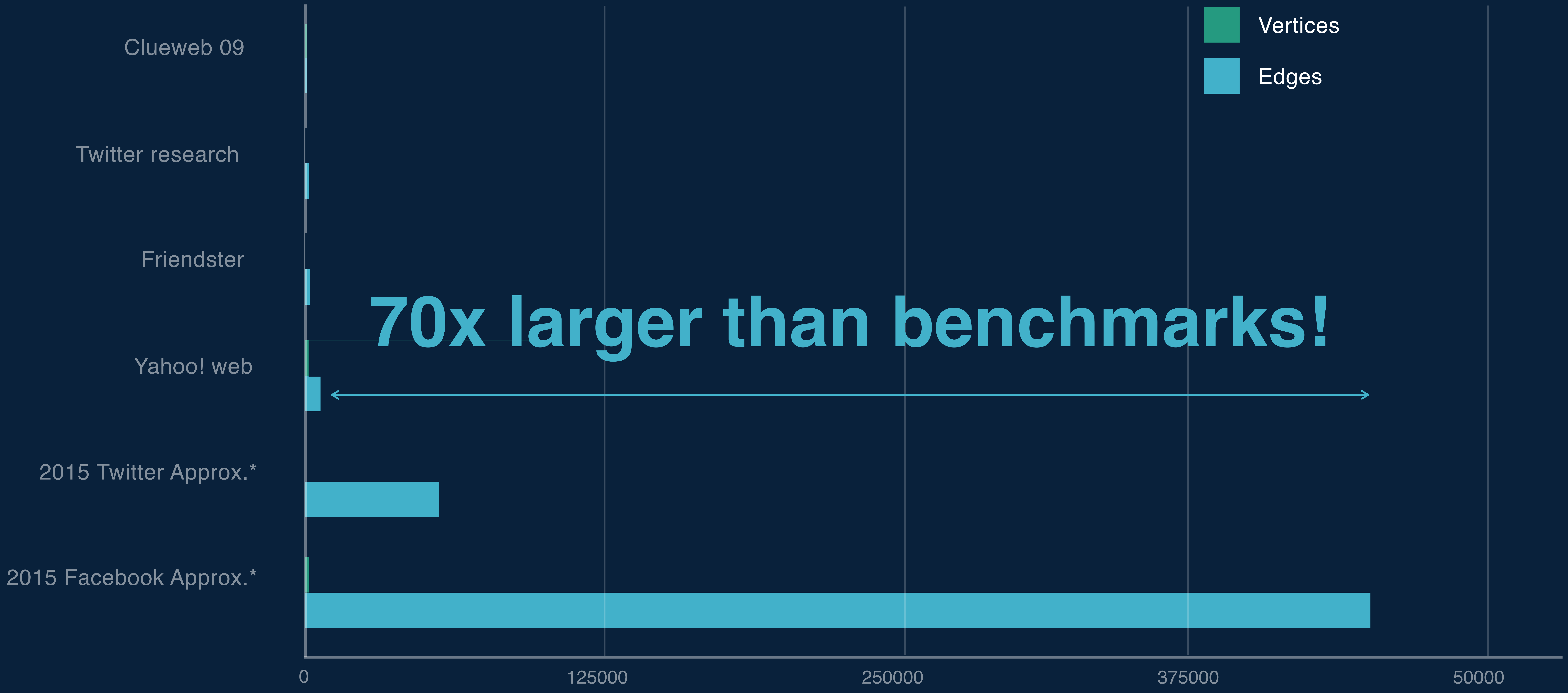
1) Capacity planning

2) Fair evaluation

Benchmark Graphs



Benchmark to Social Graphs



70x larger than benchmarks!

Existing benchmarks

graph500.org

- Kronecker graph
- Breadth First Search (BFS)

Not applicable @ FB

Algorithms

Friend of Friends counts

PageRank

Community detection

Graph partitioning

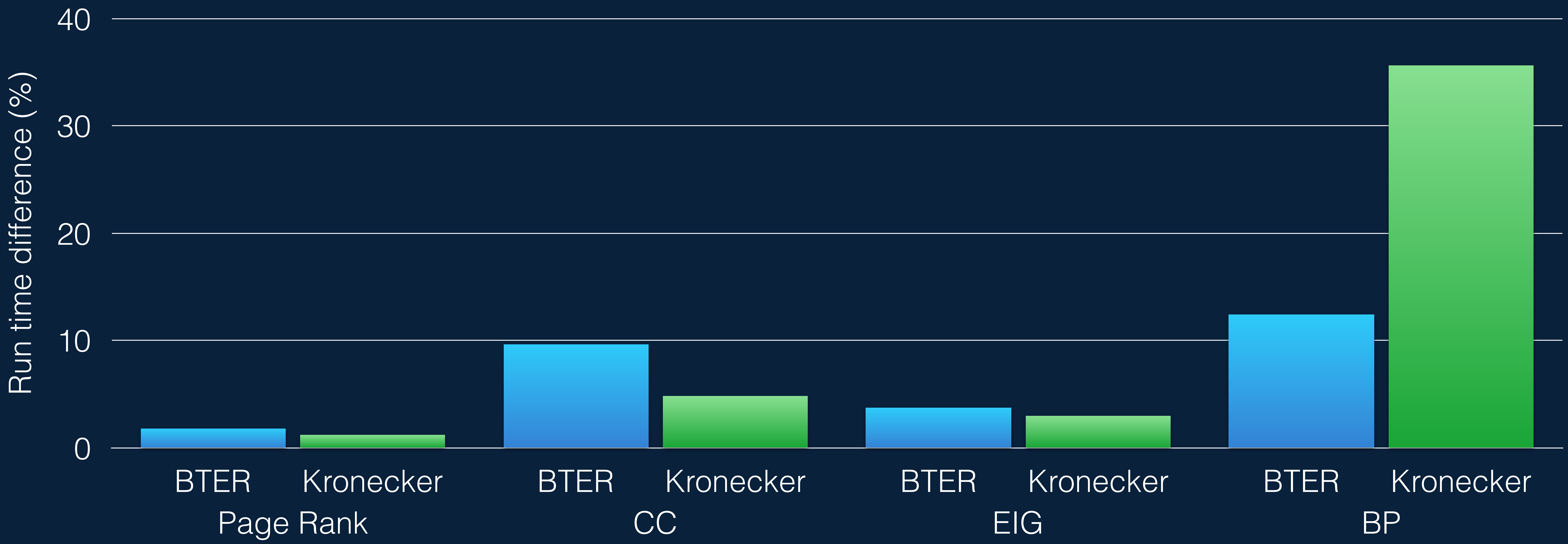
K-Core decomposition

Eigen value decomposition

Local clustering coefficient

Personalized Page Rank

Importance of fidelity



BTER: http://arxiv.org/abs/1302.6636

Known Graph Generation Algorithms

Erdos Renyi

BTER

Kronecker

R-MAT

LDBC

Random Walk

DK-2

Requirements

1. Match the graph size. If it doesn’t scale, it doesn’t work
2. Match degree distribution
3. Match joint degree and clustering coefficient (ideally dk-3 distribution)
4. Match high level application metrics

Existing algorithms vs requirements

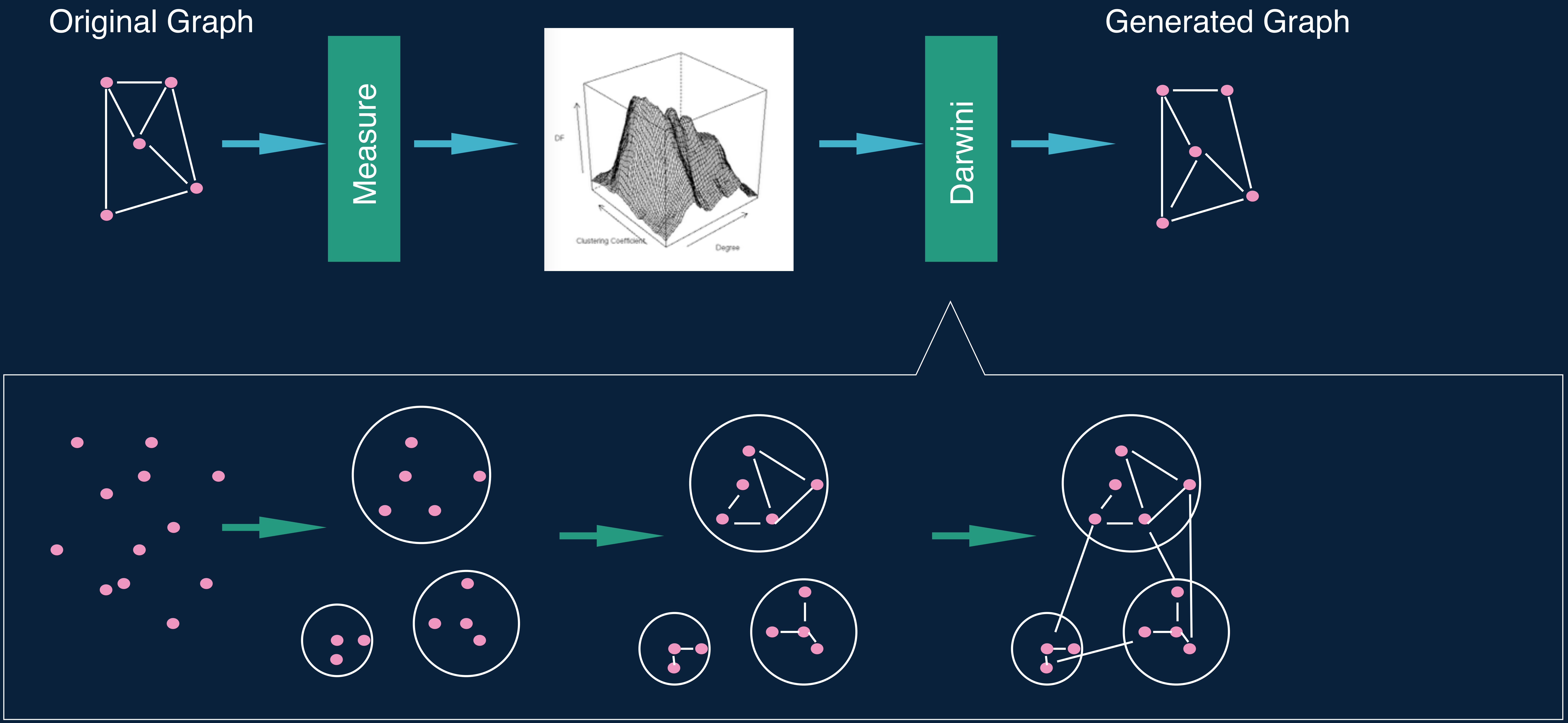
	Kronecker	BTER	Erdos-Renyi
Scalability	👍 ★	👍	👍
Degree distribution		👍	👍
Joint degree & CC			
High level metrics			

Darwini*

1. Built on Apache Giraph, scales to hundreds machines
2. Capable of generating graphs with trillions of edges
3. Generates graphs with specified joint degree-clustering coefficient distribution
4. Shows better accuracy in performance benchmarking against the original graph

*Caerostris darwini - is an orb-weaver spider that produces one of the largest known orb webs, web size ranged from 900–28000 square centimeters

Applying Darwin to the real graph



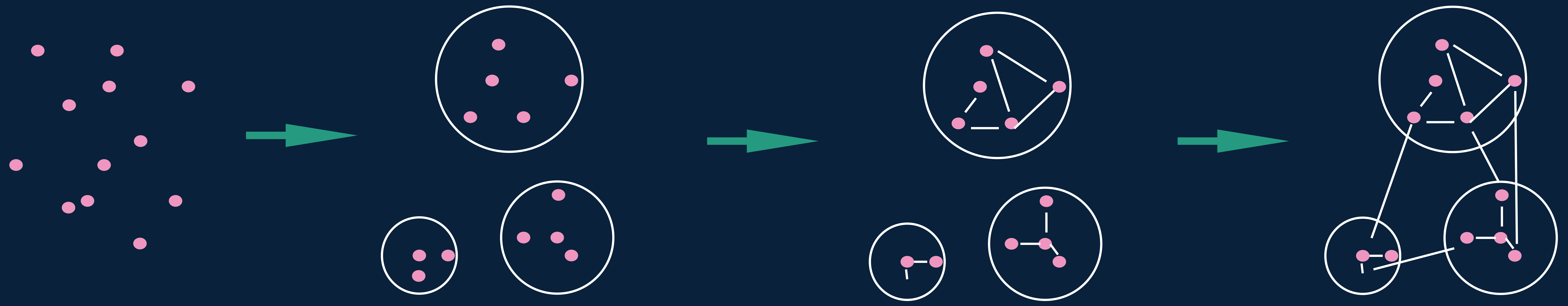
Original Graph

Measure

Darwini

Generated Graph

Darwini step by step



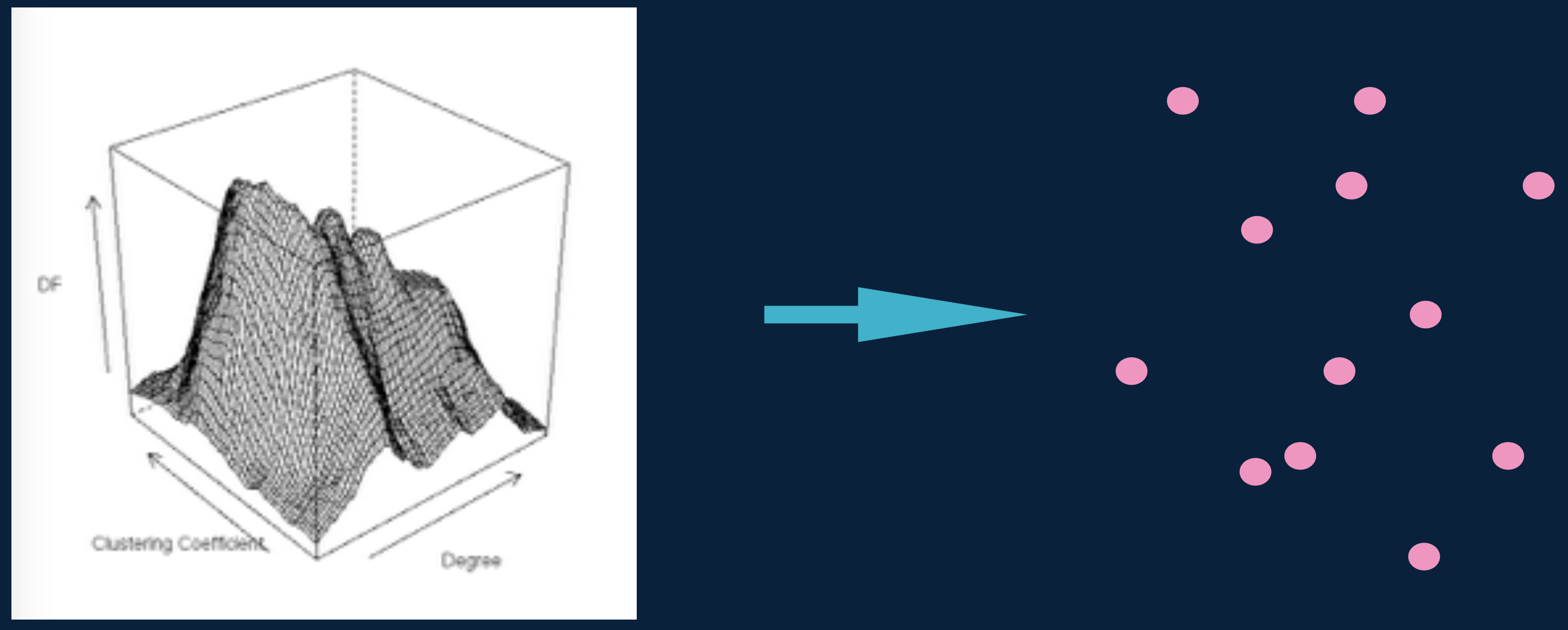
Create vertices
Assign expected degree
and clustering coefficient

Group vertices that expect
same number of triangles
together

Create random edges
within each group

Create random edges
between groups

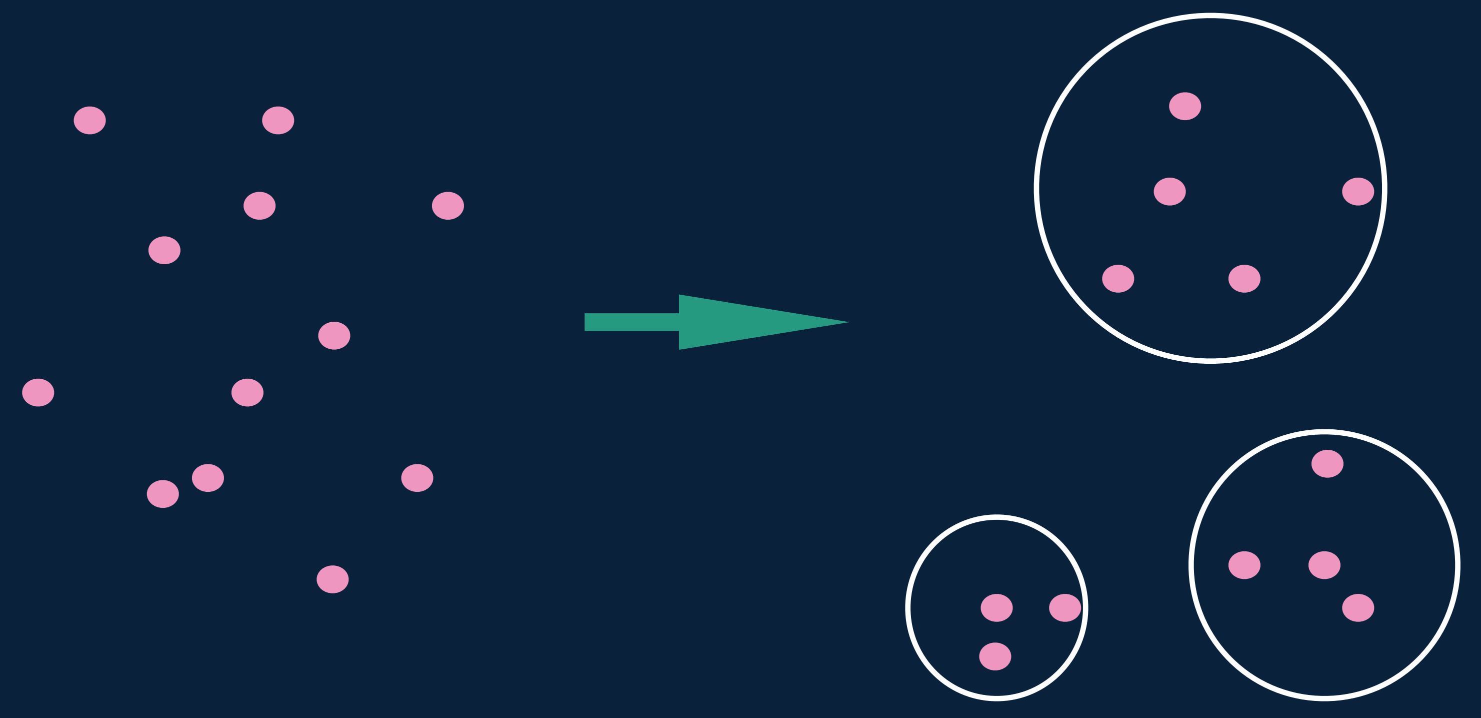
Darwini: create vertices



$\forall c_i, d_i$

Create N vertices and draw degree and clustering coefficient from the joint degre-clustering coefficient distribution

Darwini: group vertices into buckets



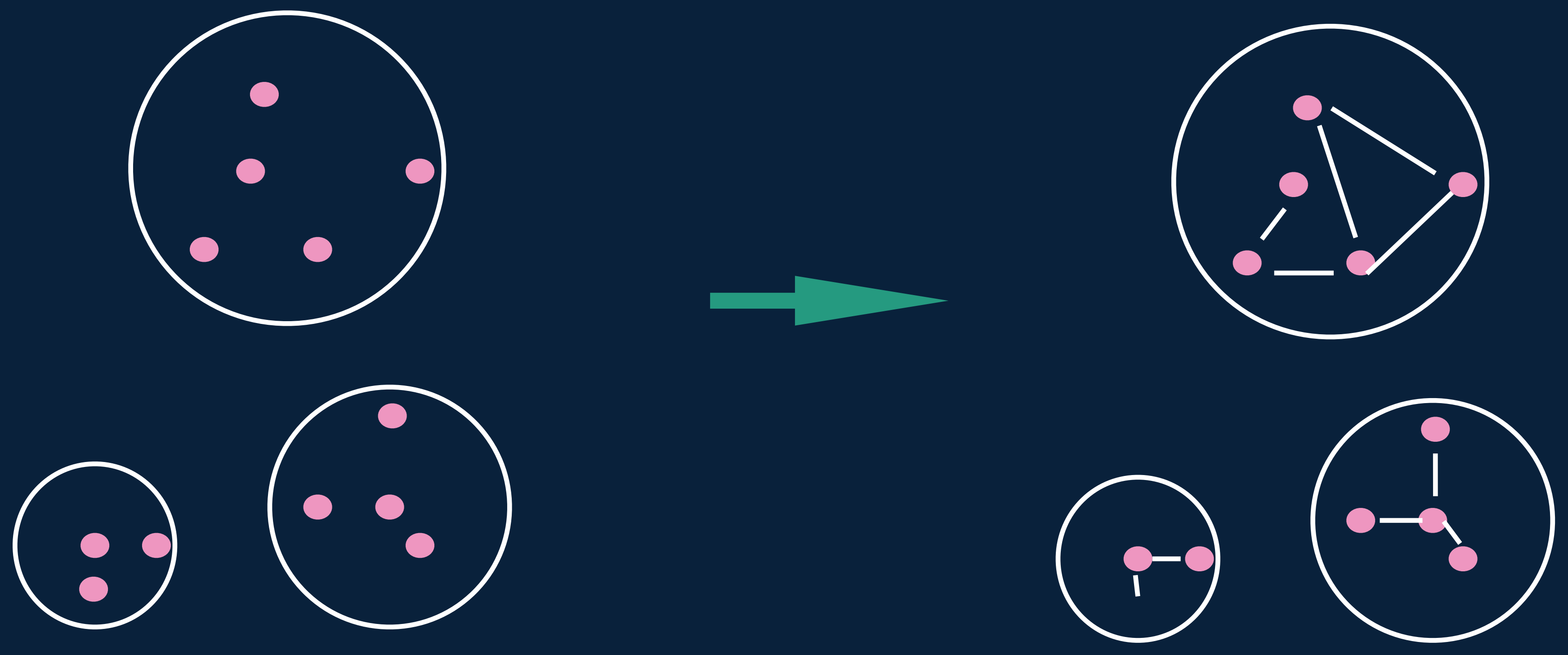
$$c_{e,i} = c_i d_i (d_i - 1)$$

Group vertices that expected to participate in the same number of triangles together

Limit the size of each bucket, so that we don't exceed expected degree

$$n \leq \min_{i \in B}(d_i) + 1 = n_{B,max}$$

Darwini: create triangles

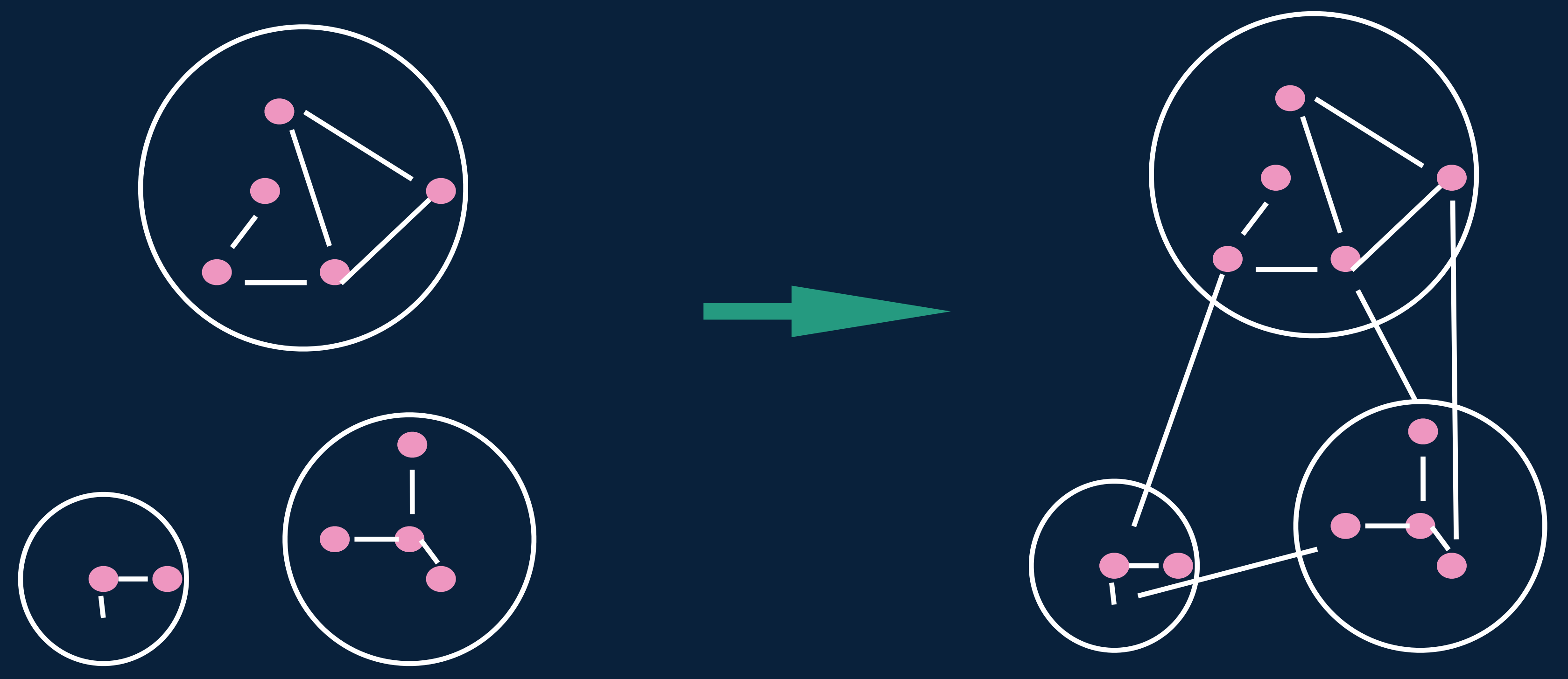


Create random edges between each pair of vertices in each bucket with probability

$$P_e = \sqrt[3]{\frac{c_i d_i (d_i - 1)}{(n-1)(n-2)}}$$

After this step, we will have enough triangles to get right clustering coefficient

Darwini: create random edges between buckets



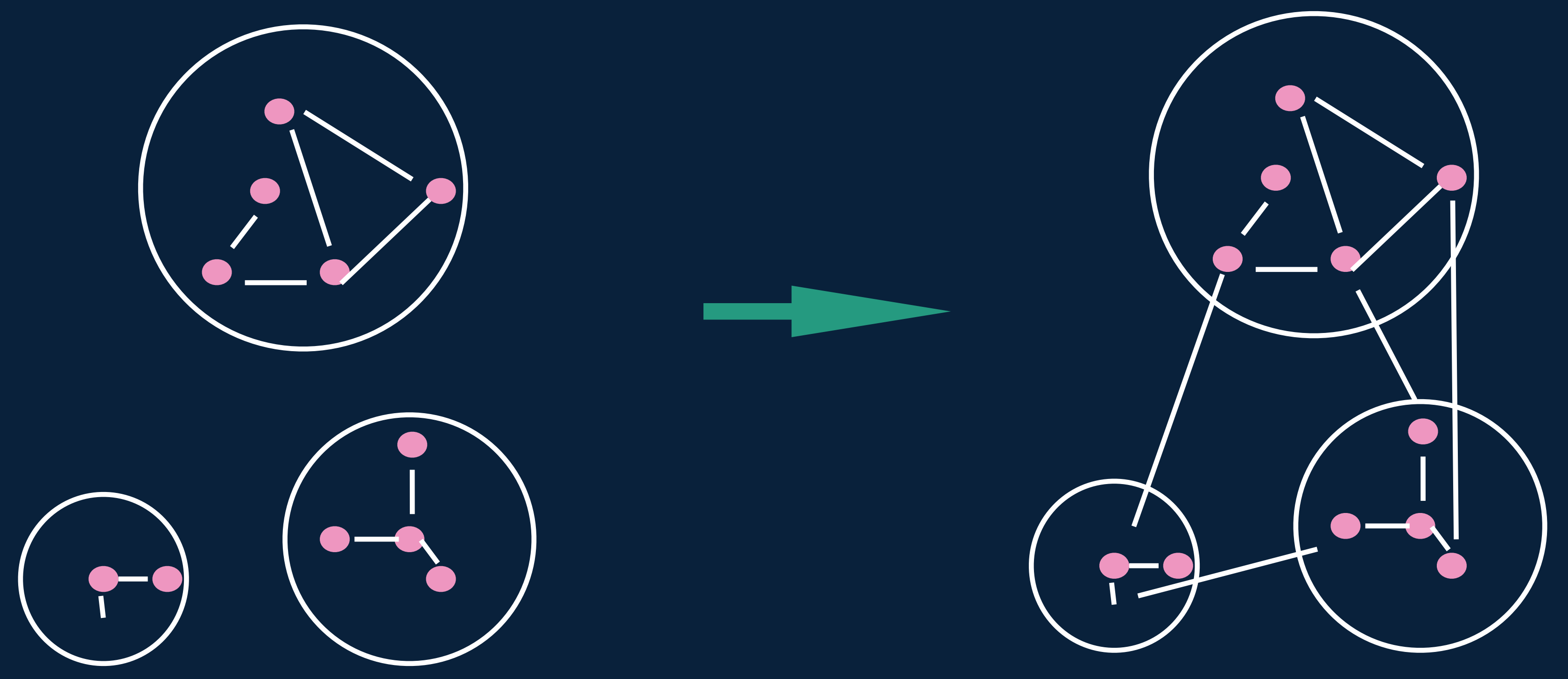
For each vertex, that doesn't have enough edges yet, pick random vertex and create an edge if another vertex doesn't have enough edges either.

Hard to find counterparts for high degree vertices

Adding random edges in Apache Giraph

1. Not all information readily available on every machine
2. Execution must be parallel
3. Exact match is not always necessary
4. Purely random connection is not enough to make realistic joint degree distribution

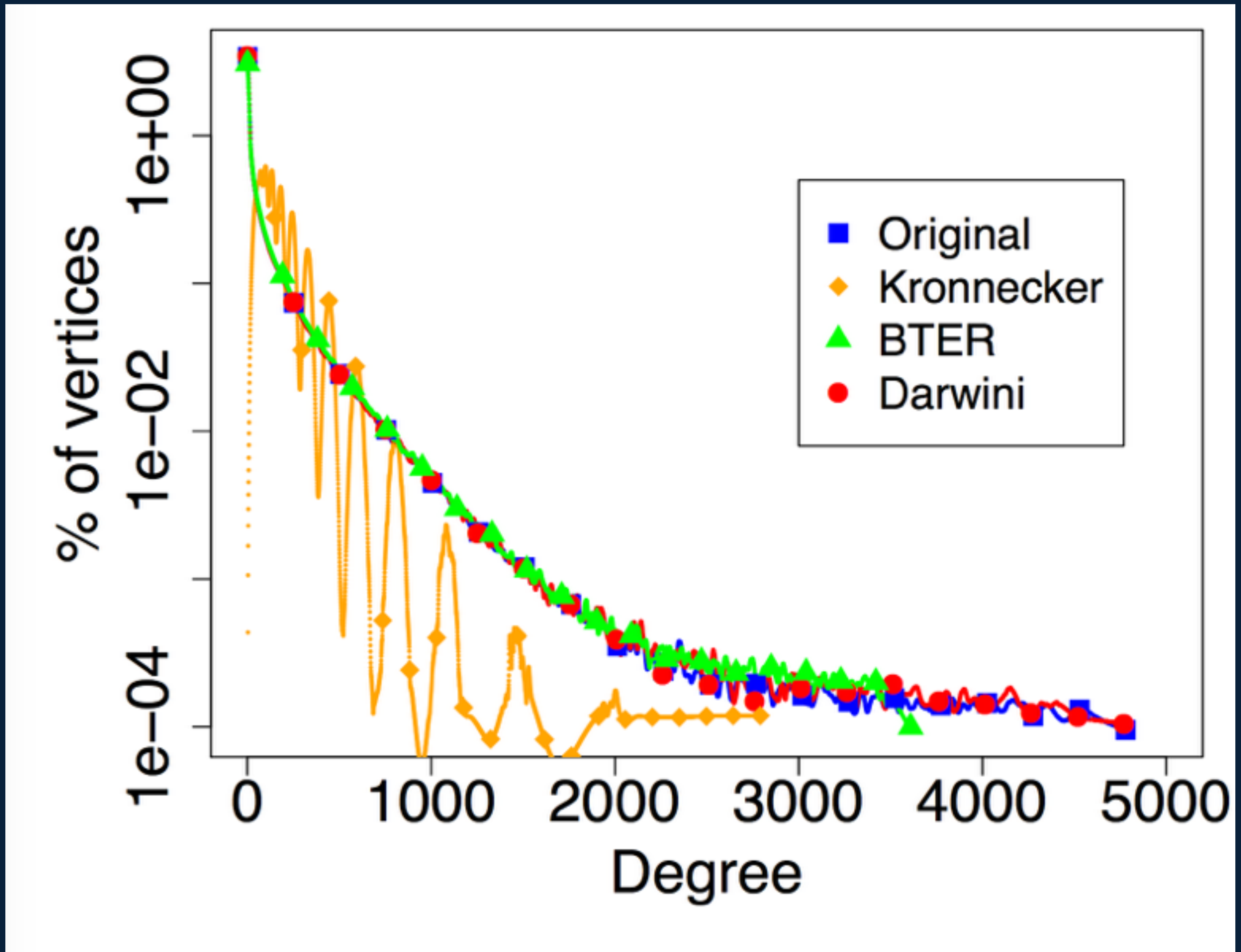
Darwini: create edges for high-degree nodes

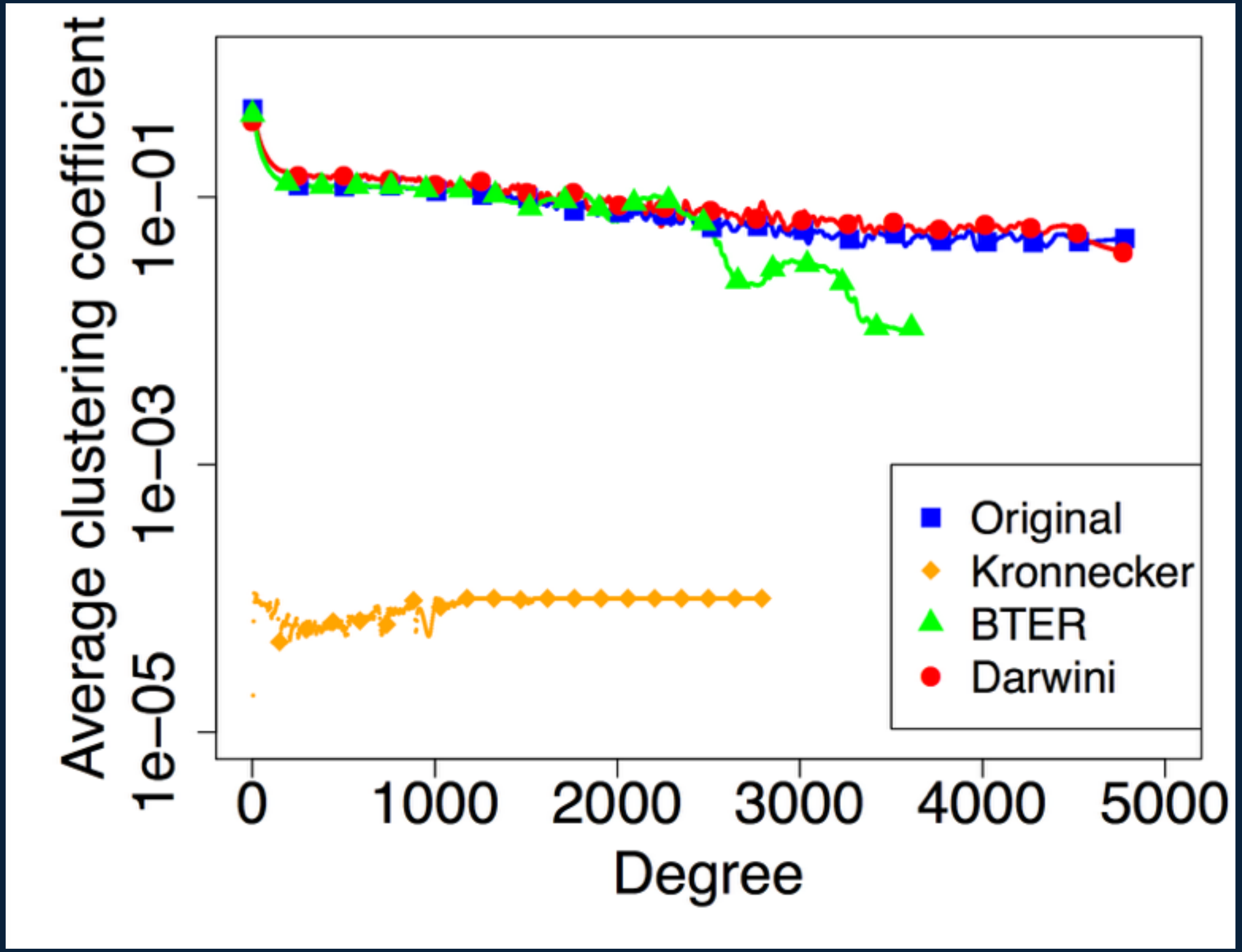


1. Group vertices into ever increasing groups.
2. For each pair of vertices within each group, connect them with probability

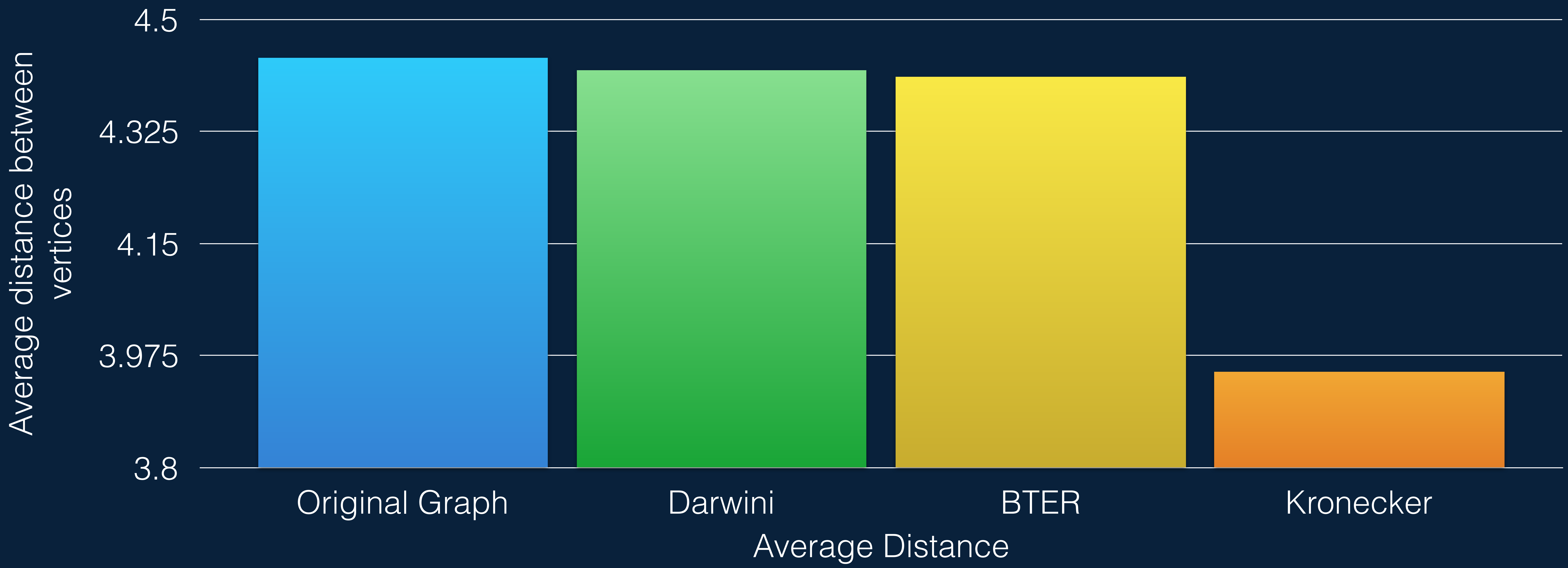
$$p = \frac{|d[i]-d[j]|}{d[i]+d[j]}$$

Results: graph quality

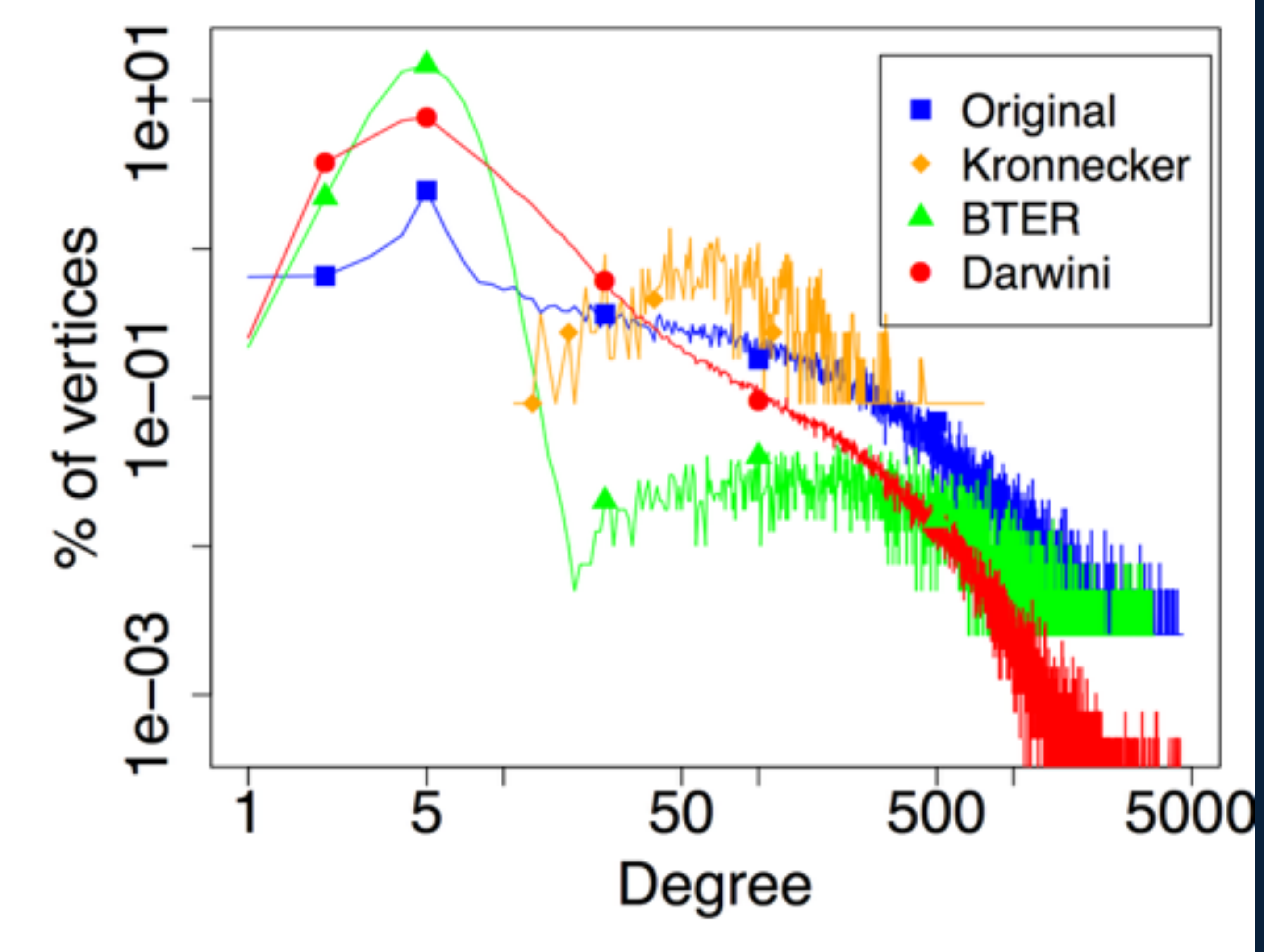


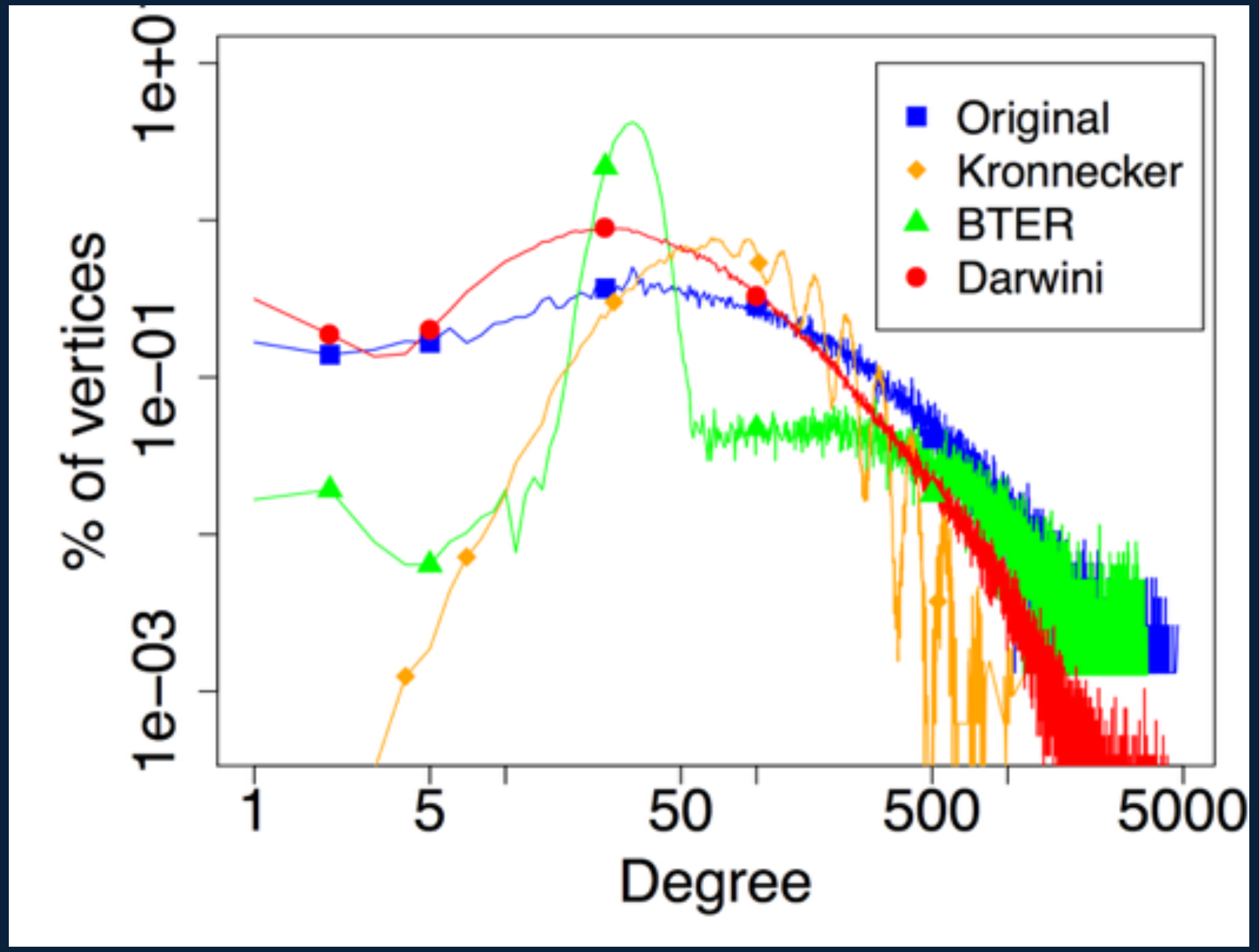


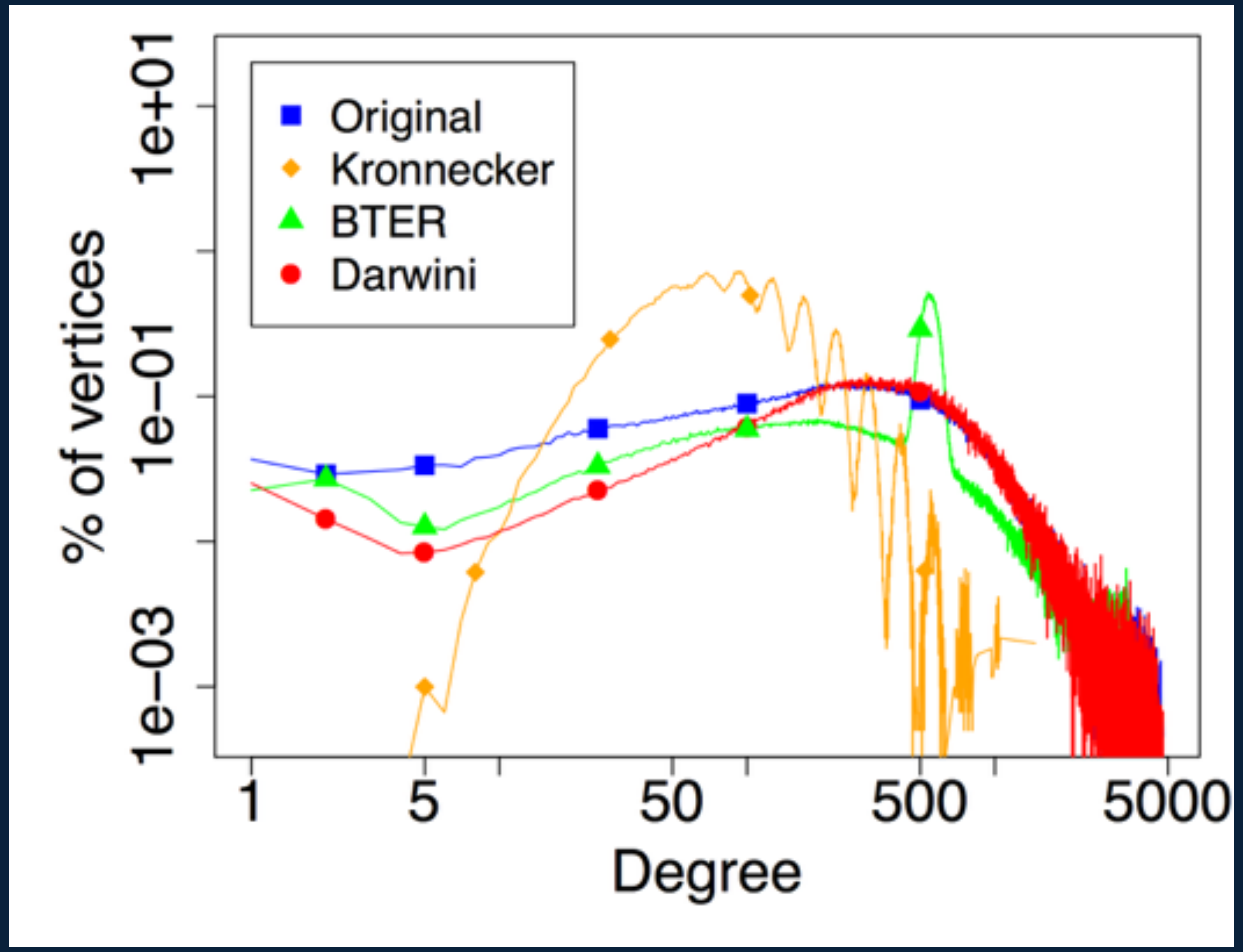
Average Distance



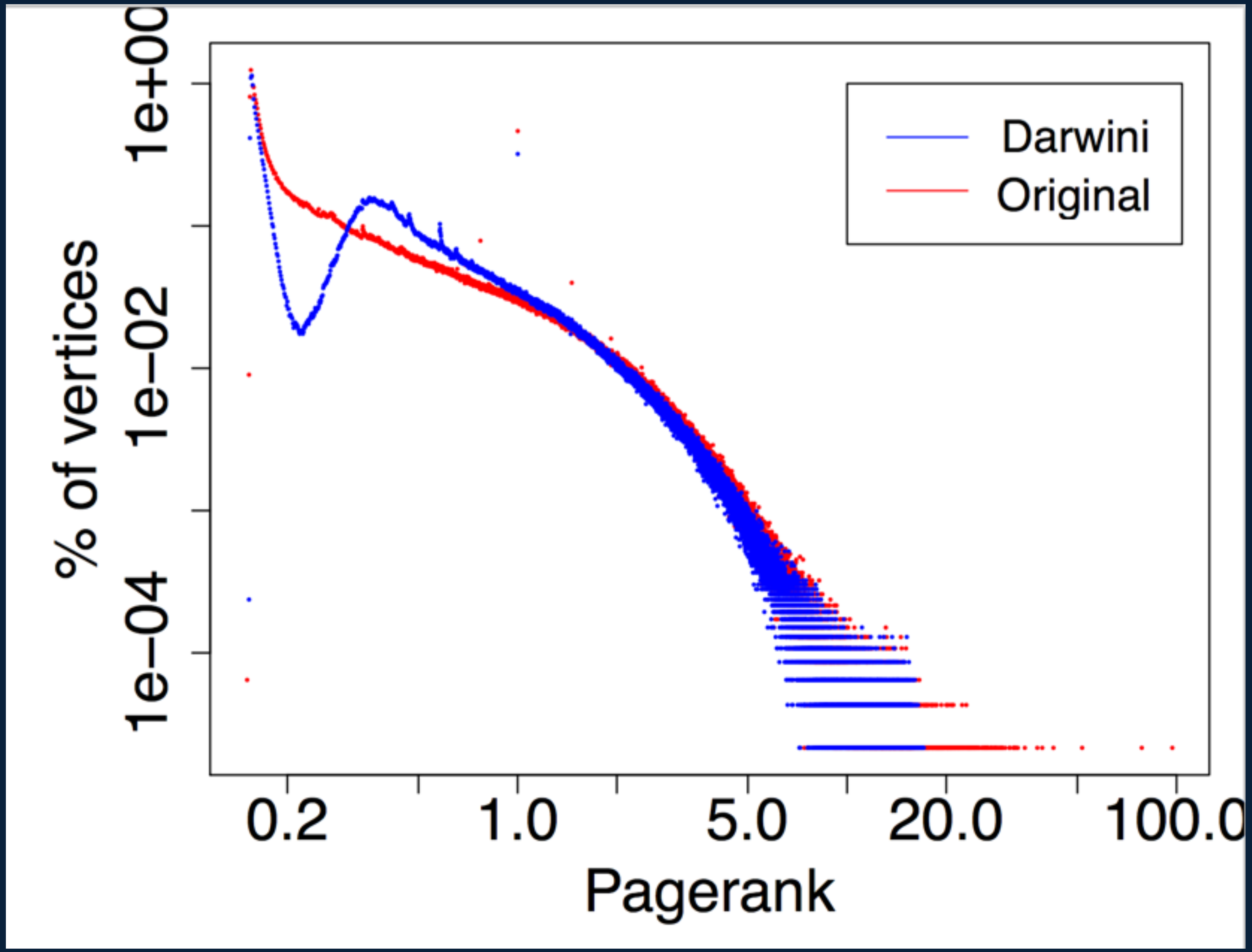
Results: joint degree distribution

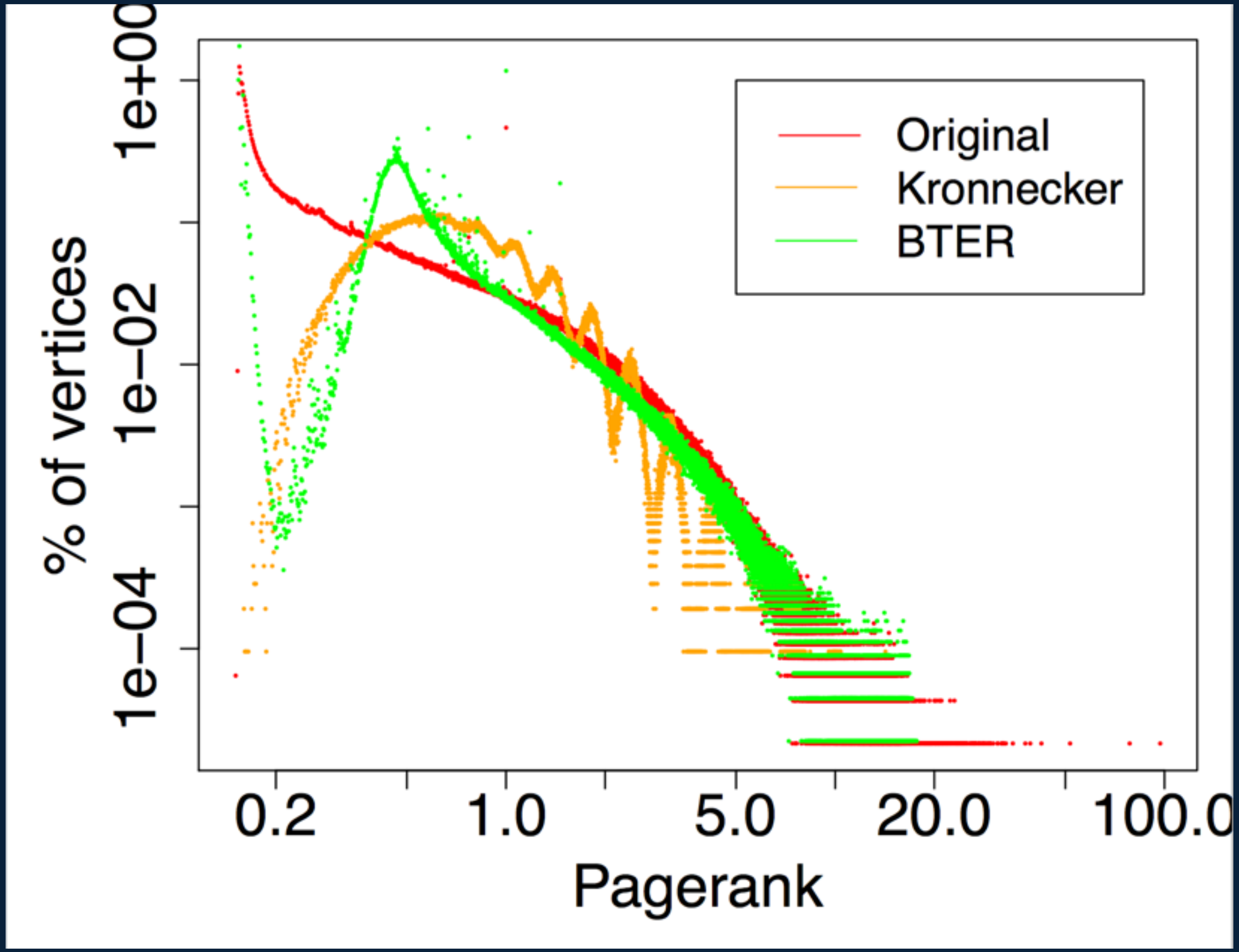




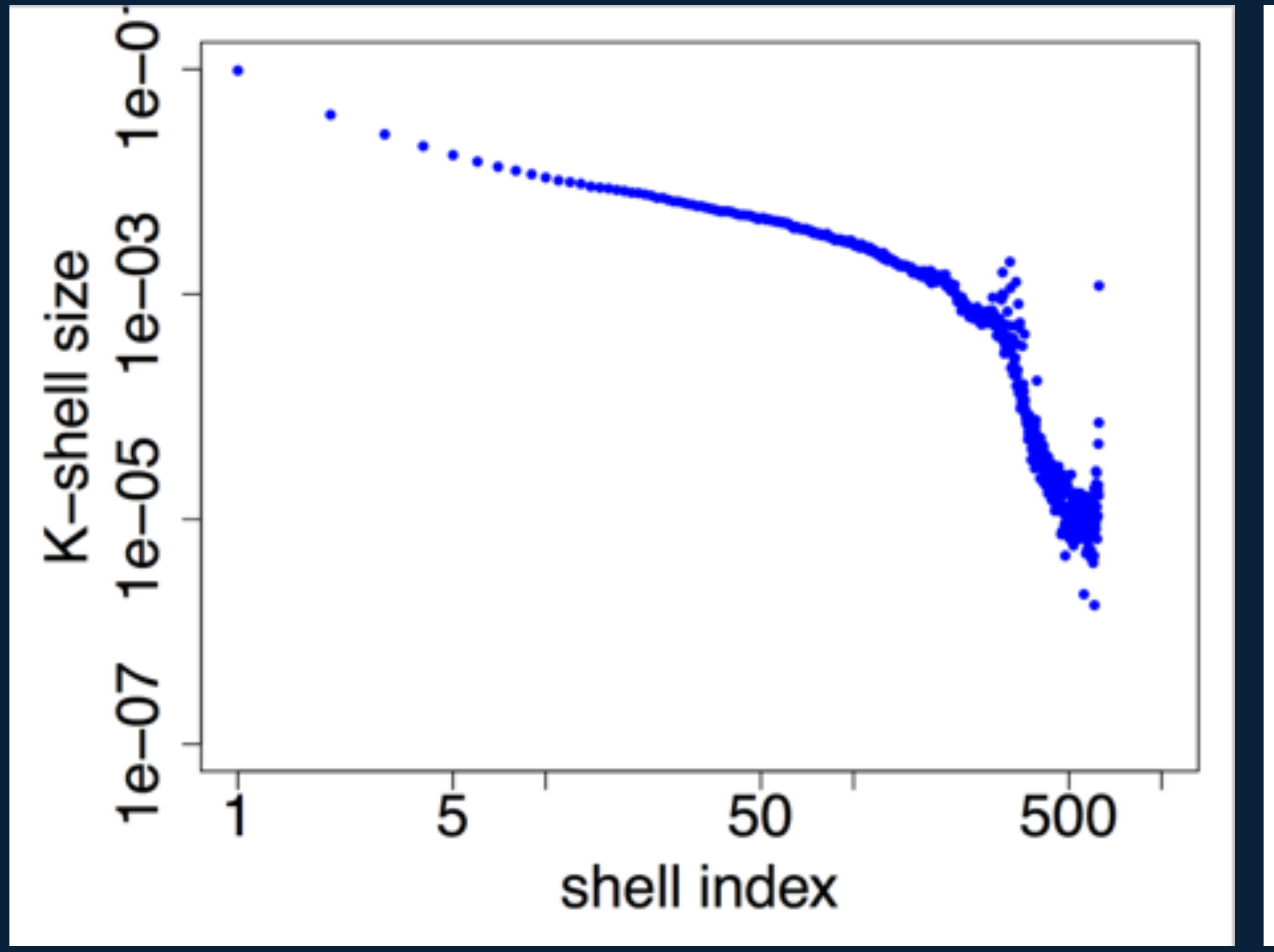


Results: page rank

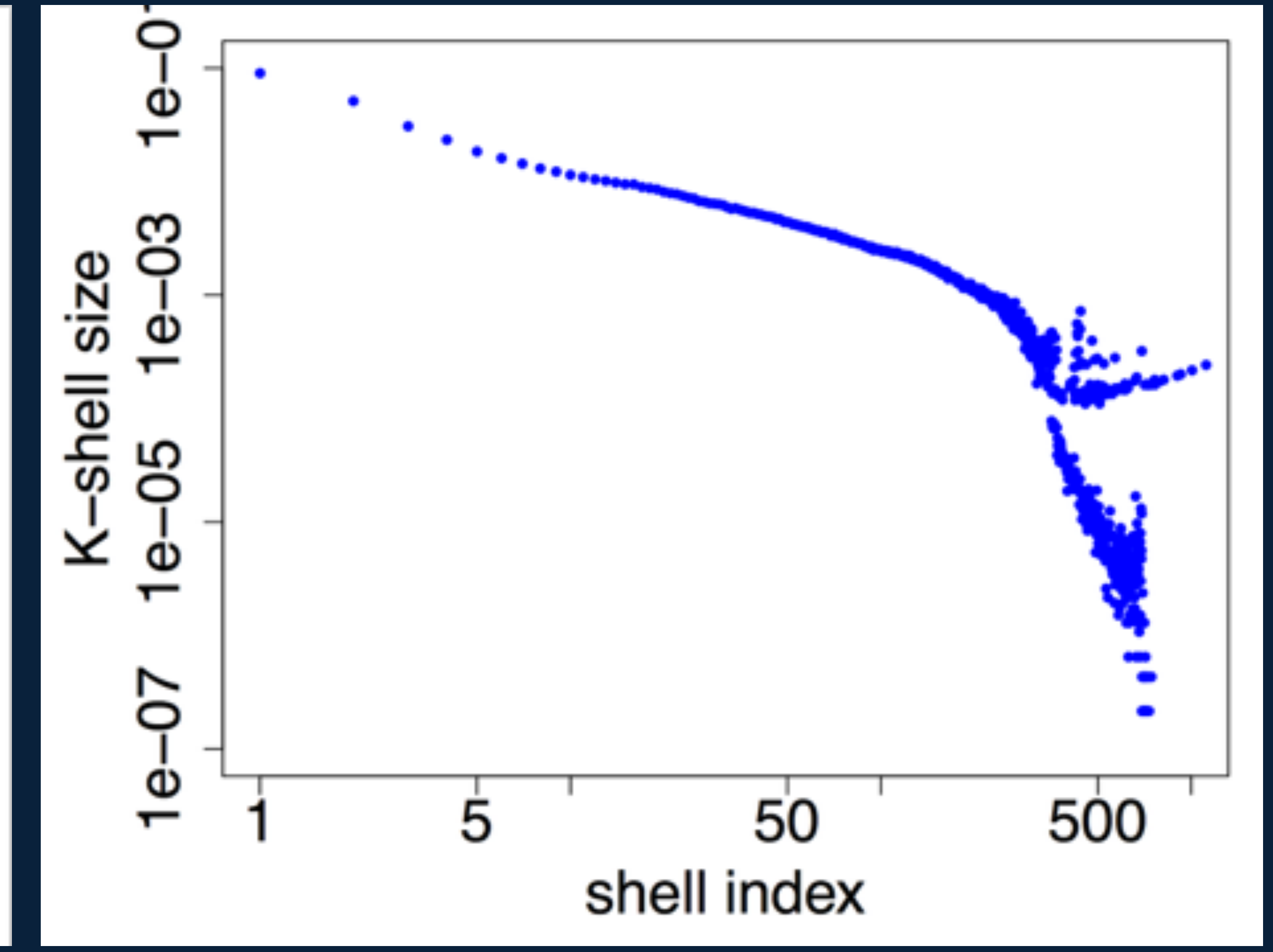




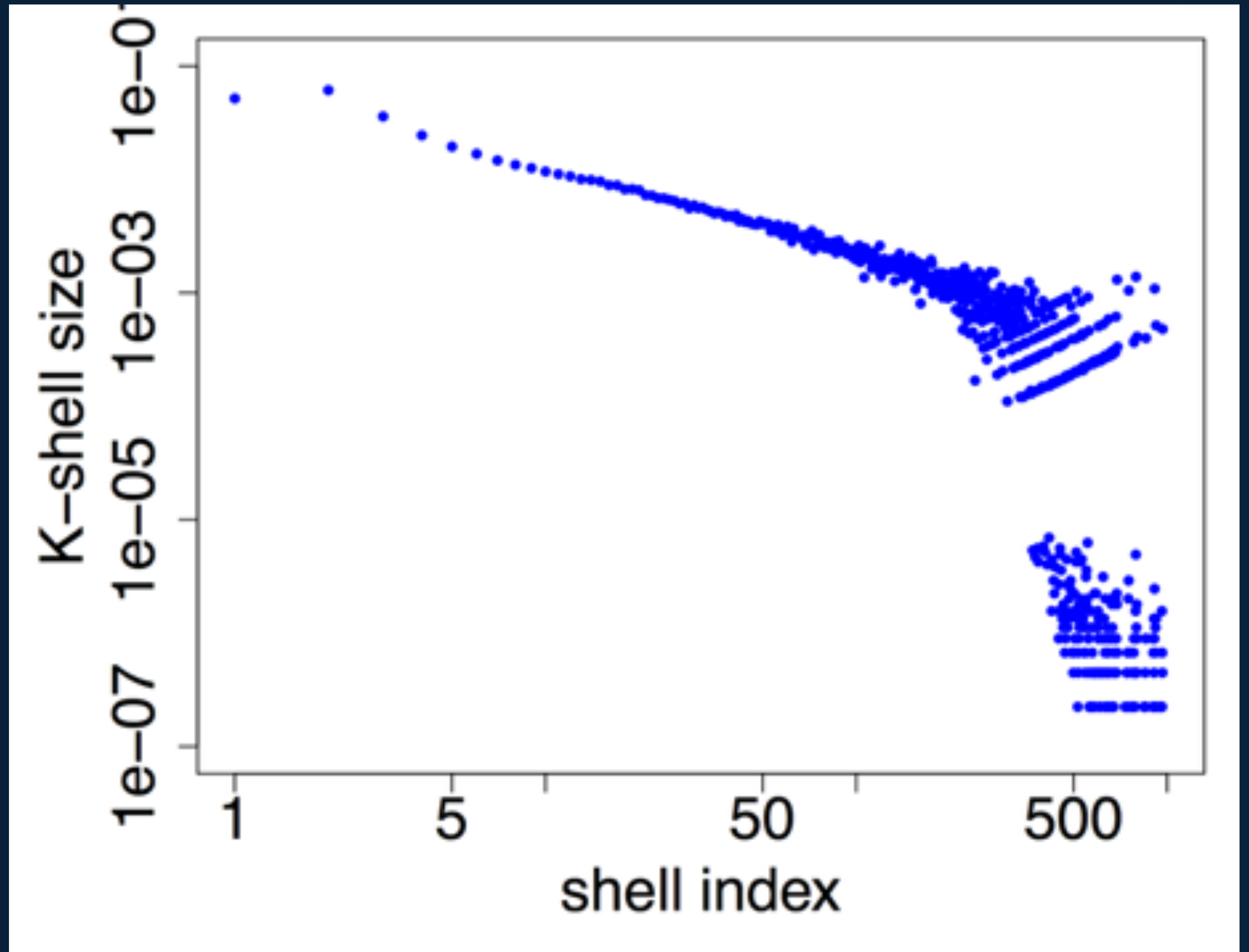
Results: K-Core decomposition



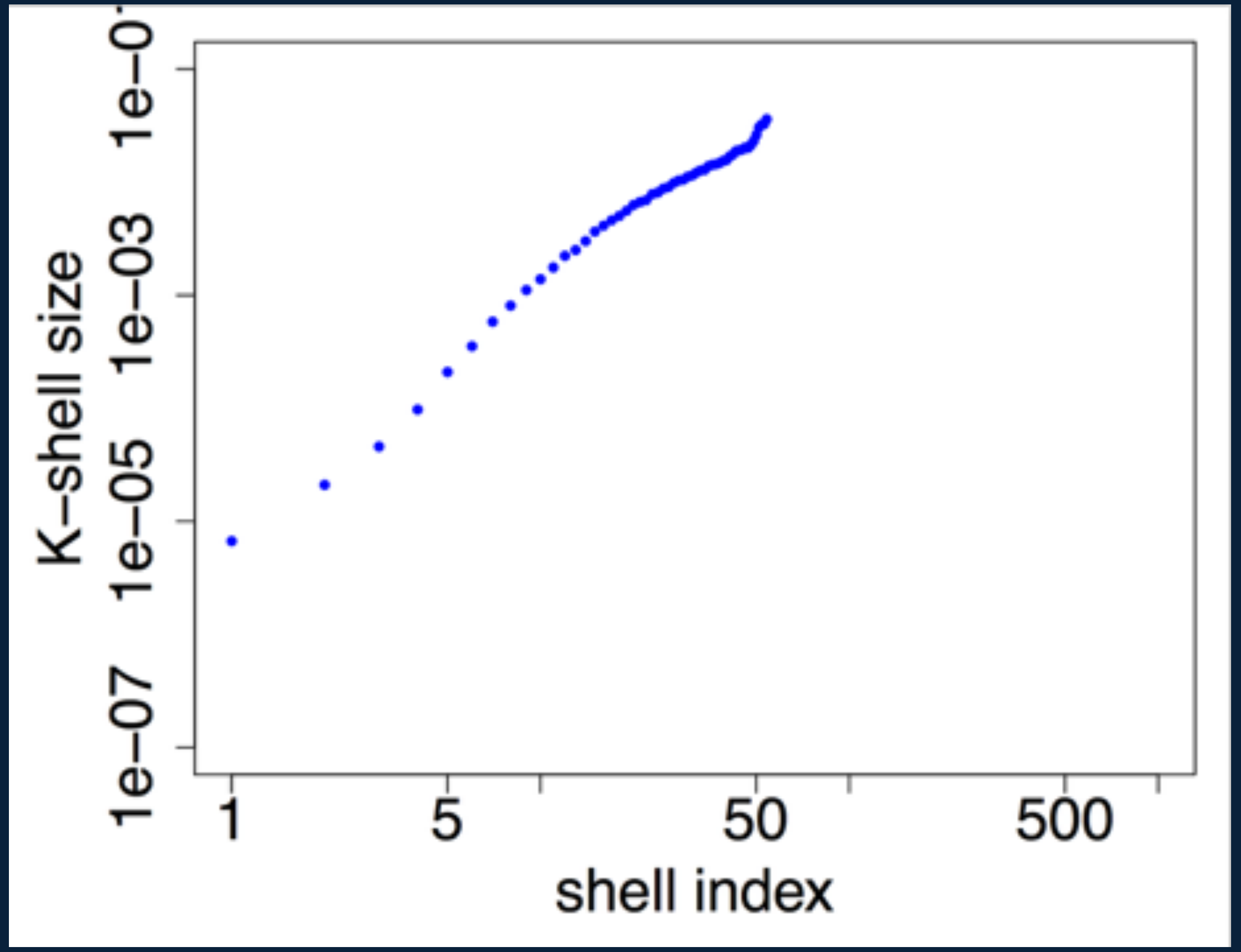
Original Graph



Darwini

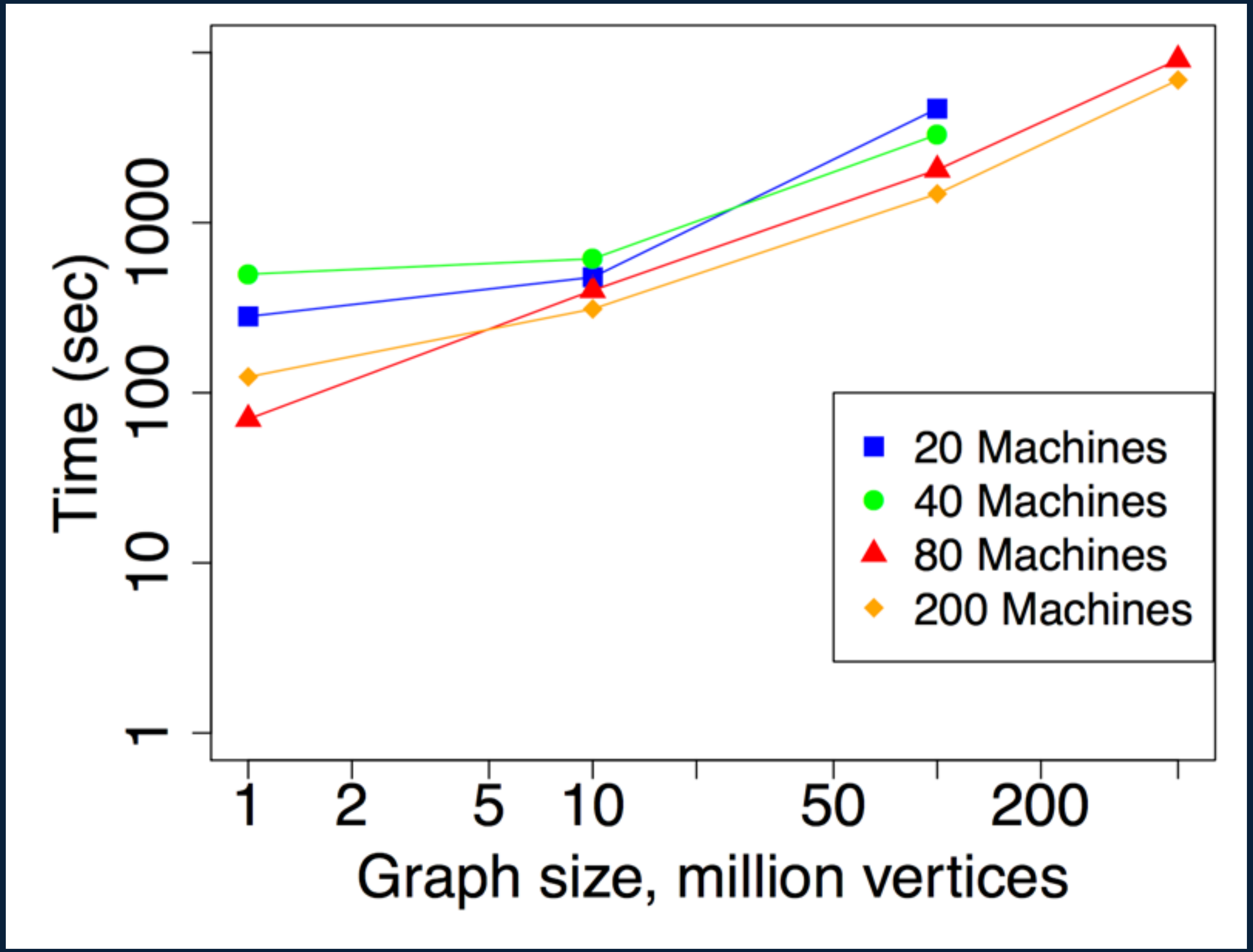


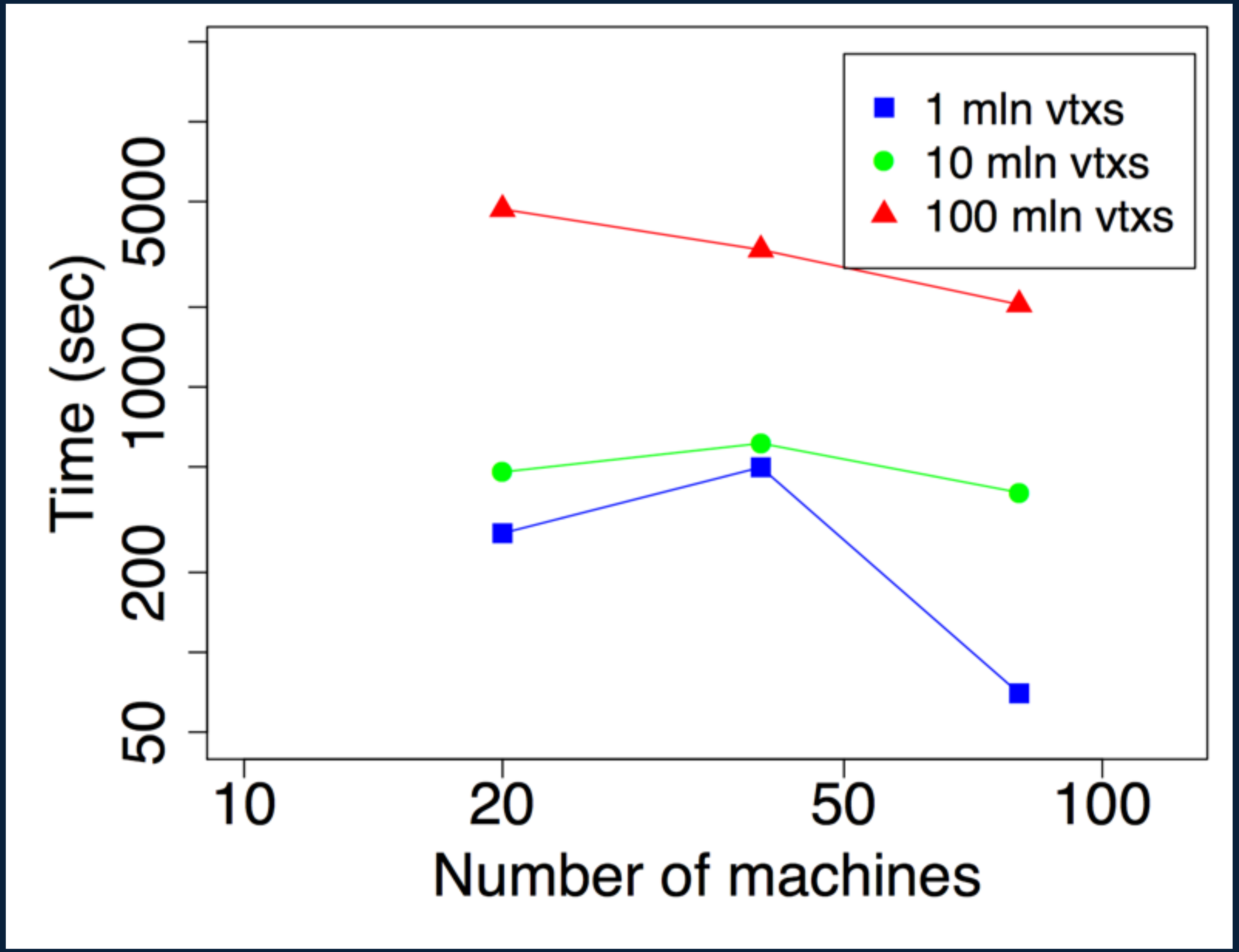
BTER



Kronecker

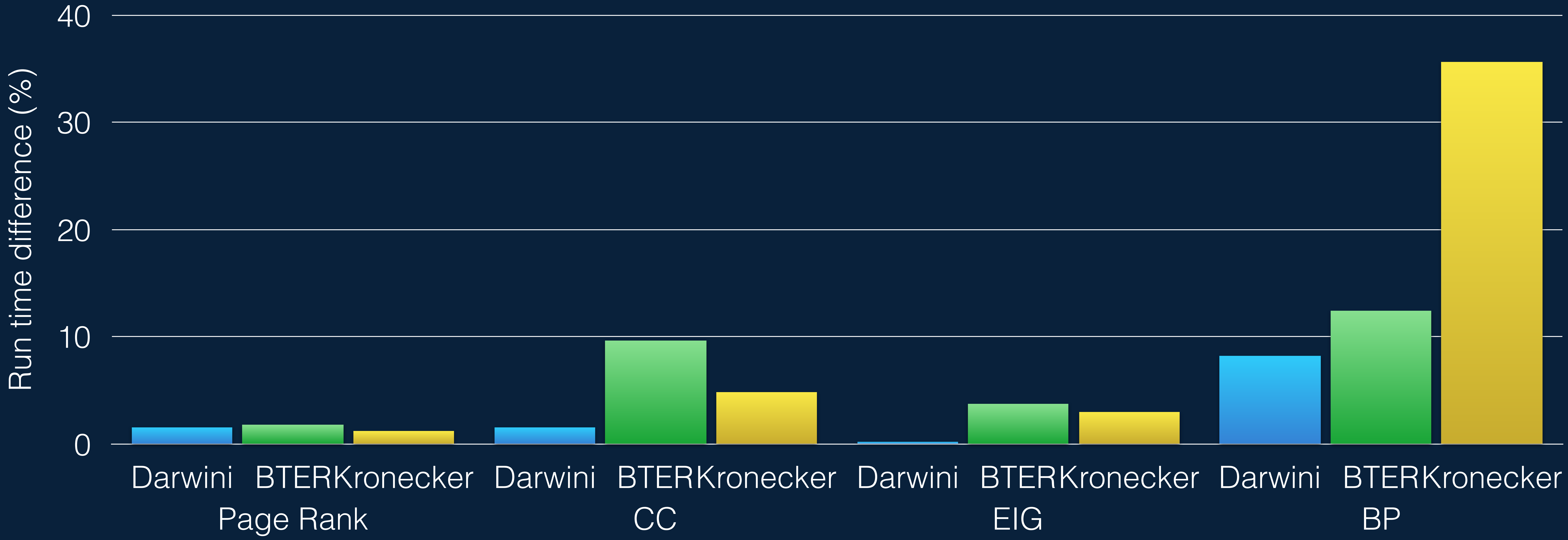
Darwini performance





Trillion edges graph in 7 hours

Results: fidelity



Thank You