2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)

Distributed Kronecker Graph Generation with Ground Truth of Many Graph Properties

Trevor Steil
School of Mathematics
University of Minnesota
Minneapolis, MN, USA
steil016@umn.edu

Benjamin Priest
Thayer School of Engineering
Dartmouth College
Hanover, NH, USA
benjamin.w.priest.th@dartmouth.edu

Geoffrey Sanders, Roger Pearce,
Timothy La Fond, Keita Iwabuchi
Center for Applied Scientific Computing (CASC)
Lawrence Livermore National Laboratory (LLNL)
Livermore, CA, USA
{sanders29, pearce7, lafond1, iwabuchi1}@llnl.gov

***Abstract*—Computing various global and local topological graph features is an important facet of data analysis. To do so robustly and scalably requires efficient graph algorithms that either calculate topological features exactly or approximate topological features accurately. For this reason researchers developing distributed graph analytic algorithms desire generated graph benchmarks that share the challenging characteristics of real-world graphs (small-world, scale-free, heavy-tailed degree distribution) with efficiently calculated ground truth to the desired ouput.**

Given two small scale-free graphs with adjacency matrices A and B, their Kronecker product graph [1] has adjacency matrix $C = A \otimes B$. Such Nonstochastic Kronecker graphs are highly compressible, and many expensive global graph calculations can be computed in sublinear time, with local graph statistics computed exactly in linear time, both from a sublinear amount of storage. Therefore, this class of graphs are likely of high interest to those pursuing data analysis tasks that incorporate diverse graph-based features.

Here, we extend previous results regarding local triangle statistics and demonstrate that ground truth Kronecker formulas apply to: (i) some distance-based vertex centrality metrics (vertex eccentricity and closeness centrality), (ii) internal and external edge density of communities. Moreover, we demonstrate several scaling laws apply that allow researchers to have control over various ground truth quantities.

I. Introduction

Graph analytics have widespread applications to many scientific fields such as computer science, biology, and social science. Computing or approximating graph analytics can be used to decorate graphs entities (vertices and edges) with a diverse class of local topological features, such as (i) centrality scores [2]–[4], (ii) pattern statistics [5], (iii) community membership [6], and (iv) roles [7]. Incorporating various local graph topolical properties as features in machine learning (ML) tasks is also an important facet of applying ML to massive datasets containing relational data.

When implementing graph algorithms, a standard technique for validation is to compare the results to a known trusted implementation. This process is useful in many cases but is not always possible. When a new problem is being solved or when new algorithms allow solving problems larger than previously possible, all validation must occur at a much smaller scale than the desired use case, which is an important step but often not satisfactory for HPC validation.

A proposed solution to this problem is to use nonstochastic Kronecker graphs as validation tools [8]. A *Kronecker graph* $\mathcal{G}_C$ has an adjacency matrix that is a Kronecker product [1], [9], [10] of two much smaller factors, $C = A \otimes B =$

$$\begin{pmatrix} A_{11}B & A_{12}B & \cdots & A_{1,n_A}B \\ A_{21}B & A_{22}B & \cdots & A_{2,n_A}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{n_A,1}B & A_{n_A,2}B & \cdots & A_{n_A,n_A}B \end{pmatrix},$$

Graph generation using nonstochastic Kronecker products can be contrasted with the stochastic Kronecker products used in the ubiquitous R-MAT generator [12]. R-MAT generators are used to produce the graphs used for various graph benchmarks, such as the Graph500 [13] and Graph Challenge [14], [15]. When using an R-MAT generator, exact graph properties cannot be determined until generation is complete. Upon generation, large graphs take very large amounts of space if they are to be stored for reuse.

The use case for nonstochastic Kronecker generators is different from that of stochastic generators, and the former will not replace the latter. The nonstochastic Kronecker generators are appropriate for validation of algorithms and generation of graphs with certain properties at different scales. The generated graphs do have some peculiar properties, such as the lack of vertices with large prime degrees. Stochastic generators are appropriate for fast generation of graphs with certain properties, in expectation.

Several previous works demonstrate useful Kronecker formulas and bounds for ground truth graph statistics, including degree distribution, triangle distribution, graph diameter, and eigenvalues [8], [11], [16], [17]. This paper extends these works by deriving efficient formulas for additional ground truth quantities that can be computed for nonstochastic Kronecker products. In general, for a graph with $|\mathcal{E}_C|$ edges, suppose a desired graph analytic $f(C)$ costs $O(|\mathcal{E}_C|^p)$. If a simple Kronecker formula of the form

$$f(C) = \sum_s (g_s(A) \otimes h_s(B))$$

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, and was reviewed and released as LLNL-CONF-768148.

978-1-5386-5555-9/19/$31.00 ©2019 IEEE
DOI 10.1109/IPDPSW.2019.00048

with a low number of terms exists, then a data structure requiring $O(|\mathcal{E}_C|^{p/2})$ storage can produce ground truth with $O(|f(C)| + |\mathcal{E}_C|^{p/2})$ cost. This means global scalar quantities (such as a global triangle count) are computed sublinearly, in $O(|\mathcal{E}_C|^{p/2})$ time, and local quantities (such as triangle counts at edges) are produced in linear time.

Viewing these ground truth quantities from a slightly different perspective, we can generate graphs with desired characteristics at different scales. We summarize several important scaling laws in the table below (note that the assumptions on A and B sometimes differ slightly to attain the presented law).

Quantity	Scaling Law
Vertices	$n_C = n_A n_B$
Edges	$m_C = 2m_A m_B$
Degree	$\mathbf{d}_C = \mathbf{d}_A \otimes \mathbf{d}_B$
Vertex Triangles	$\mathbf{t}_C = 2\mathbf{t}_A \otimes \mathbf{t}_B$
Edge Triangles	$\Delta_C = \Delta_A \otimes \Delta_B$
Global Triangles	$\tau_C = 6\tau_A\tau_B$
Clustering Coeff.	$\eta_C(p) \gtrapprox \frac{1}{3}\eta_A(i)\eta_B(k)$
Vertex Eccentricity	$\epsilon_C(p) = \max\{\epsilon_A(i), \epsilon_B(k)\}$
Graph Diameter	$\max\{\text{diam}(\mathcal{G}_A), \text{diam}(\mathcal{G}_B)\}$
# Communities	$\lvert\Pi_C\rvert = \lvert\Pi_A\rvert\lvert\Pi_B\rvert$
Internal Density	$\rho_{in}(C) \gtrapprox \frac{1}{3}\rho_{in}(A)\rho_{in}(B)$
External Density	$\rho_{out}(C) \lessapprox 4\rho_{out}(A)\rho_{out}(B)$

Our contributions are summarized as follows:

(a) We provide an open-source distributed asynchronous implementation that reads two factor graphs A and B from file and efficiently produces the nonstochastic Kronecker graph $C = A \otimes B$.
(b) We extend the results in [11] to derive Kronecker formulas for vertex and edge triangle participation in the case of self loops on every vertex in the factors. These results yield linear computation of ground truth local triangle counts from a sublinear amount of memory.
(c) We derive scaling laws for vertex and edge clustering coefficients that demonstrate vertex clustering coefficients are controllable, yet edge clustering coefficients are not.
(d) We derive Kronecker formulas and scaling laws for graph distance, diameter, eccentricity, and closeness centrality. These formulas yield linear computation of ground truth from a sublinear amount of memory for every vertex in all 4 of these analytics except for closeness centrality, where a subset of closeness centrality scores can be computed efficiently from a compressed format.
(e) We derive Kronecker formulas and scaling laws for internal/external community edge counts and edge density, which are both controllable, under reasonable assumptions.
(f) We discuss several advantages and disadvantages we have observed regarding using nonstochastic Kronecker graphs as various classes of benchmarks for massive-scale graph analytics.

II. Preliminaries

Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a set of $n := |\mathcal{V}|$ vertices and $|\mathcal{E}|$ edges, pair-wise relationships between members of $\mathcal{V}$ of the form $(i, j) \in \mathcal{E}$, where $i, j \in \mathcal{V}$. We say $\mathcal{G}$ is *undirected* if $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$ for every (i, j) (and $\mathcal{G}$ is *directed* if this doesn't hold for a single edge). An edge of the form $(i, i) \in \mathcal{E}$ is a *self loop*.

Let $\mathbb{B} = \{0, 1\}$. The matrix $A \in \mathbb{B}^{n\times n}$ is an *adjacency matrix* representing $\mathcal{G}$ if $A_{ij} = 1$ for each $(i, j) \in \mathcal{E}$ and $A_{ij} = 0$ for each $(i, j) \notin \mathcal{E}$. Given an adjacency matrix A, we use $\mathcal{G}_A$, $\mathcal{V}_A$, and $\mathcal{E}_A$, to represent the associated graph, vertices, and edges, respectively. Additionally, we use a subscript A for many other symbols referring to properties of $\mathcal{G}_A$ (e.g. $n_A = |\mathcal{V}_A|$).

A. Algebraic Properties of Kronecker Products

Matrices formed by Kronecker products are block structured and we define some convenience functions to write the index maps compactly. For a block-structured array with block-size n, we define functions that, for a given *global index* i, retrieve the *block number*, $\alpha_n(i)$, and the *intra-block index* $\beta_n(i)$.

$$\begin{aligned} \alpha_n(i) &= \lfloor (i-1)/n \rfloor + 1, \\ \beta_n(i) &= [(i-1)\%n] + 1. \end{aligned}$$

We drop the subscript n when it is clear from context. The inverse of $i \to (\alpha_n(i), \beta_n(i))$ is

$$\gamma_n(x, y) = (x-1)n + y,$$

in the sense that $i = \gamma_n(\alpha_n(i), \beta_n(i))$.

Def. 1. (Kronecker Product [1], [9], [10]) *Let $A \in \mathbb{R}^{m_A\times n_A}$ and $B \in \mathbb{R}^{m_B\times n_B}$. The* Kronecker Product *of A and B is $(A \otimes B) \in \mathbb{R}^{(m_A m_B)\times(n_A n_B)}$ and has entries*

$$(A \otimes B)_{pq} = \left(A_{\alpha_{m_B}(p),\alpha_{n_B}(q)}\right)\left(B_{\beta_{m_B}(p),\beta_{n_B}(q)}\right)$$

for $1 \le p \le (m_A m_B)$ and $1 \le q \le (n_A n_B)$, or, equivalently,

$$(A \otimes B)_{\gamma_{m_B}(i,k),\gamma_{n_B}(j,l)} = A_{ij}B_{kl},$$

for $1 \le i \le m_A, 1 \le j \le n_A, 1 \le k \le m_B$, and $1 \le l \le n_B$.

Prop. 1. (Properties of the Kronecker Product [1], [9], [10])

(a) Scalar Multiplication. *For any $a_1, a_2 \in \mathbb{R}$,*

$$(a_1 a_2)(A_1 \otimes A_2) = (a_1 A_1) \otimes (a_2 A_2).$$

(b) Distributivity.

$$\begin{aligned} (A_1 + A_2) \otimes A_3 &= (A_1 \otimes A_3) + (A_2 \otimes A_3) \quad \textit{and} \\ A_1 \otimes (A_2 + A_3) &= (A_1 \otimes A_2) + (A_1 \otimes A_3). \end{aligned}$$

(c) Transposition. $(A_1 \otimes A_2)^t = (A_1^t \otimes A_2^t)$.
(d) Matrix-Matrix Multiplication. *When $n_{A_1} = m_{A_3}$ and $n_{A_2} = m_{A_4}$,*

$$(A_1 \otimes A_2)(A_3 \otimes A_4) = (A_1 A_3) \otimes (A_2 A_4).$$

Def. 2. (Haddamard Product [10]) *Let $A, B \in \mathbb{R}^{m\times n}$. The* Haddamard Product *of A and B is $(A \circ B) \in \mathbb{R}^{m\times n}$, with*

$$(A \circ B)_{ij} = A_{ij}B_{ij}$$

for $1 \le i \le m$ and $1 \le j \le n$.

Def. 3. (Standard Matrix and Vector Objects) *Given $A \in \mathbb{R}^{n_A\times n_A}$, O_A is the* matrix of all zeros *and I_A is the* identity matrix*, both with the same size as A. Constant vectors $\mathbf{0}_A, \mathbf{1}_A \in \mathbb{R}^{n_A}$, are the* vector of all zeros*, and the* vector of all ones*, both of dimension n_A.*

We define some diagonal operators of square matrices in terms of the Haddamard product so it is transparent in Kronecker formulas we derive.

Def. 4. (Matrix Diagonal Operators and Self Loops) *Given $A \in \mathbb{R}^{n_A\times n_A}$, the matrix $D_A = I_A \circ A$ is the* diagonal entries *of A. The* diagonal operator *is diag$(A) := (I_A \circ A)\mathbf{1}_A$, a vector in $\mathbb{R}^{n_A}$. Diagonal entries of D_A that are nonzero represent* self loops *in $\mathcal{G}_A$, when $D_A = I_A$ we say A has full self loops, and when $D_A = O_A$ we say A has no self loops.*

Now we recall several useful formulas regarding Haddamard products.

Prop. 2. (Properties of the Haddamard Product [10]) *In the following, we implicitly assume that $n_A = n_B$ and $m_A = m_B$ whenever $A \circ B$ is present.*

(a) COMMUTATIVITY. $A_1 \circ A_2 = A_2 \circ A_1$.

(b) SCALAR MULTIPLICATION. *For any $a_1, a_2 \in \mathbb{R}$,*

$$(a_1a_2)(A_1 \circ A_2) = (a_1A_1) \circ (a_2A_2).$$

(c) DISTRIBUTIVITY.

$$\begin{aligned}(A_1 + A_2) \circ A_3 &= (A_1 \circ A_3) + (A_2 \circ A_3) \quad \text{and}\\ A_1 \circ (A_2 + A_3) &= (A_1 \circ A_2) + (A_1 \circ A_3).\end{aligned}$$

(d) TRANSPOSITION. $(A_1 \circ A_2)^t = (A_1^t \circ A_2^t)$.

(e) HADDAMARD-KRONECKER DISTRIBUTIVITY.

$$(A_1 \otimes A_2) \circ (A_3 \otimes A_4) = (A_1 \circ A_3) \otimes (A_2 \circ A_4).$$

(f) DIAGONAL-KRONECKER DISTRIBUTIVITY. *When $m_{A_1} = n_{A_1}$ and $m_{A_2} = n_{A_2}$,*

$$\mathit{diag}(A_1 \otimes A_2) = \mathit{diag}(A_1) \otimes \mathit{diag}(A_2).$$

III. AN HPC IMPLEMENTATION

In this section, we present our Kronecker generator for use in HPC applications. To generate the Kronecker product $C = A \otimes B$, we replicate B across all processors. For our purposes, we assume A and B are given as (unordered) edge lists. Edges of A are evenly distributed across the R processors. Let $A_r \in \mathbb{R}^{n_A\times n_A}$ denote the adjacency matrix containing only the edges in A assigned to processor r. Then processor r is responsible for generating

$$C_r = A_r \otimes B,$$

and $C = \sum_r C_r$. This generation process requires $\mathcal{O}(|\mathcal{E}_A|R^{-1} + |\mathcal{E}_B|)$ storage on each processor and $\mathcal{O}(|\mathcal{E}_A||\mathcal{E}_B|R^{-1})$ time to generate the product graph. Replicating A and B across all processors is not ideal, but these adjacency matrices are typically very small relative to the size of C. For instance, if $|\mathcal{E}_A| \approx |\mathcal{E}_B|$, then $|\mathcal{E}_C| \approx 2|\mathcal{E}_A|^2$.

If edges are being stored, the processor responsible for generating an edge must then send it to the processor responsible for its storage as determined by some mapping scheme. If A and B were sorted and placed in a compressed sparse row (CSR) structure, it would be possible for a processor to efficiently generate only the edges it must store in some cases, but this would be dependent on the method used to distribute edges to processors. We make the generation of Kronecker edges independent of the storage method of edges so the approach is more modular.

Source code for the Kronecker generator is available as part of HavoqGT [18], LLNL's asynchronous graph library. Very recently this approach was used to generate a trillion-edge graph (whose factors where two Graph500 scale 18 graphs with different random seeds) for validation purposes within a CORAL2 benchmark. It was generated in under a minute on 1.57M cores of IBM BG/Q SEQUOIA at LLNL.

Rem. 1. (SCALING) *This implementation has been useful for our purposes of generating billion- to trillion-edge graphs with ground truth on thousands to millions of processors. However, it has a scalability problem and would not weakly scale to 10^{15} edge graphs on many millions of processors, if the factors A and B are similar in number of edges. Assume we are generating C with $|\mathcal{E}_C|$ from A and B with $|\mathcal{E}_A|$, $|\mathcal{E}_B| = \mathcal{O}(|\mathcal{E}_C|^{1/2})$. Because only the edges of A are distributed, we can make use of only $\mathcal{O}(|\mathcal{E}_C|^{1/2})$ processors.*

A simple solution with our current implementation is to fix the size of B and let $|\mathcal{E}_A|$ grow proportionally to $|\mathcal{E}_C|$.

A more general solution is given by distributing the edges of both A and B. Let $R_{1/2} := \lceil R^{1/2} \rceil$ and partition the edges of A into $R_{1/2}$ parts and the edges of B into $\lceil R/R_{1/2} \rceil$ parts. Having processor r generate

$$C_r = A_{r\%R_{1/2}} \otimes B_{\lfloor r/R_{1/2} \rfloor},$$

we would use $\mathcal{O}(|\mathcal{E}_C|)$ processors and weak scaling is possible.

IV. TRIANGLE PARTICIPATION

In this section we show new formulas for how many triangles reside at vertices and edges in the case where every vertex has a self loop in both factors. Then we demonstrate scaling laws for vertex and edge clustering coefficients. Lastly, we discuss the ability to create subgraphs of Kronecker graphs where ground truth triangle counts can be calculated in a highly reliable fashion, which is a useful contribution for a good-faith benchmark where the full Kronecker structure would be less exploitable (either explicitly or implicitly).

A. All Self Loops in Both Factors

In previous work [11], we derived formulas for triangle participation at vertices/edges with self loops on any vertex in a single factor ($D_A \neq O_A$, but $D_B = O_B$), and extended these results to the many types of directed graphs and labeled graphs. Self loops in A provide the user with the ability to locally tune triangle counts somewhat, while the design choice of no self loops in B leaves fairly simple Kronecker formulas (with a few terms). Here, we consider the important case where both factors have self loops at every vertex, in order to create the densest structure possible for Kronecker graphs.

1) Vertex Participation:

Def. 5. (Triangle Participation at Vertices) *For adjacency matrix A,* triangle participation at vertices *is represented by $\mathbf{t}_A \in \mathbb{R}^{n_A}$, a vector that counts the number of undirected triangles at each vertex. For A undirected with self loops, we have*

$$\mathbf{t}_A := \frac{1}{2} diag\Big((A - A \circ I_A)^3\Big).$$

Note that when A has no self loops, then $\mathbf{t}_A = \frac{1}{2} diag(A^3)$. The triangle participation at vertex $i \in \mathcal{V}_A$ is $t_i^{(A)}$, or t_i when relation to A is clear from context.

We derive a variant of Thm. 1 from [11] and the proof is in the appendix.

Cor. 1. (Triangles at Vertices with Full Self Loops) *Let $A \circ I_A = O_A$, $B \circ I_B = O_B$. Let both factors have self loops for all vertices added in $C = (A + I_A) \otimes (B + I_B)$. The number of triangles incident to $p \in \mathcal{V}_C$ is*

$$t_p = 2t_i t_k + 3(t_i d_k + d_i d_k + d_i t_k) + t_i + t_k,$$

where t_i (or t_k) and d_i (or d_k) are the triangle count and degree, respectively, for vertex $i \in \mathcal{V}_A$ (or $k \in \mathcal{V}_B$).

2) Edge Participation:

Def. 6. (Triangle Participation at Edges) Triangle participation at edges*, is a $n_A \times n_A$ matrix*

$$\Delta_A := (A - A \circ I_A) \circ (A - A \circ I_A)^2,$$

whose (i,j)-th entry is the number of triangles in which edge (i,j) participates. When A has no self-loops, $\Delta_A = (A \circ A^2)$. The triangle participation at edge $(i,j) \in \mathcal{E}_A$ is $\Delta_{ij}^{(A)}$, or Δ_{ij} when relation to A is clear from context.

We derive a variant of Thm. 2 from [11] and the details of the proof are also in the appendix.

Cor. 2. (Triangles at Edges with Full Self Loops) *Let $A \circ I_A = O_A$, $B \circ I_B = O_B$. Let both factors have self loops for all vertices added in $C = (A + I_A) \otimes (B + I_B)$. The number of triangles incident to $(p,q) \in \mathcal{E}_C$ is $\Delta_{pq} =$*

$$\begin{gathered}\Delta_{ij}\Delta_{kl} + 2(\Delta_{ij} + \Delta_{kl}) \\ +\Delta_{ij}(d_k + 1)\delta(k,l) + \Delta_{kl}(d_i + 1)\delta(i,j) \\ +2(d_i\delta(i,j) + d_k\delta(k,l) + 1),\end{gathered}$$

where Δ_{ij} (or Δ_{kl}) and d_i (or d_k) are the triangle count at edge $(i,j) \in \mathcal{E}_A$ (or $(k,l) \in \mathcal{E}_B$) and degree for vertex $i = \alpha(p) \in \mathcal{V}_A$ (or $k = \beta(p) \in \mathcal{V}_B$), respectively, and $\delta(a,b) = 1$ if $a = b$ and 0 otherwise.

B. Clustering Coefficients

Clustering coefficients of vertices and edges are useful topological characterizations of graph neighborhoods [19]. Essentially, they are the ratio of observed triangles to the maximum possible triangles, given the vertex degree(s), and range from 0 (star- or tree-like neighborhood) to 1 (clique-like neighborhood). We show the Kronecker formulas imply scaling laws, which are only controlled in the case of vertex clustering coefficients.

Def. 7. Clustering Coefficient [19] *The* clustering coefficient *at vertex i in $\mathcal{G}_A$ is*

$$\eta_A(i) = \frac{2t_i}{d_i(d_i - 1)}.$$

The clustering coefficient *at edge (i,j) in $\mathcal{G}_A$ is*

$$\xi_A(i,j) = \frac{\Delta_{ij}}{\min\{d_i, d_j\} - 1}$$

Thm. 1. *Let both factors have no self loops, or $I_A \circ A = O_A$, $I_B \circ B = O_B$. Consider any $p \in \mathcal{V}_C$, with $i = \alpha(p)$ and $k = \beta(p)$ such that $t_i, t_k > 0$ and $d_i, d_j \geq 2$. The vertex clustering coefficients p satisfy*

$$\eta_C(p) = \theta_p \, \eta_A(i) \, \eta_B(k),$$

where

$$\theta_p := \frac{(d_i - 1)(d_k - 1)}{d_i d_k - 1},$$

which is in the interval $[1/3, 1)$.

Proof. $\eta_C(p) =$

$$\begin{aligned} &= \frac{2t_p}{d_p(d_p - 1)} = \frac{4t_i t_k}{d_i d_k (d_i d_k - 1)} \\ &= \frac{(d_i - 1)(d_k - 1)}{d_i d_k - 1} \left(\frac{2t_i}{d_i(d_i - 1)}\right)\left(\frac{2t_k}{d_k(d_k - 1)}\right) \\ &= \theta_p \, \eta_A(i) \, \eta_B(k). \end{aligned}$$

The minimum possible value of θ_p is $1/3$ and is realized for $d_i = d_k = 2$, where θ_p is monotonically increasing as either d_i or d_k increase, getting arbitrarily close 1 for large values. □

Thus a controlled scaling law with vertex clustering coefficients exists. The values of θ_p are even closer to 1 if self loops are added to either or both factors. In fact $\theta_p = 1$ is possible when self loops are in both factors and both $\eta_A(i) = \eta_B(k) = 1$ (consider cliques for A and B).

Although a scaling law also exists for edge clustering coefficients, there is not a controlled bound.

Thm. 2. *Let both factors have no self loops, or $I_A \circ A = O_A$, $I_B \circ B = O_B$. Consider any $(p,q) \in \mathcal{G}_C$, with $i = \alpha(p)$, $k = \beta(p)$, $j = \alpha(q)$, $l = \beta(q)$, such that $\Delta_{ij}, \Delta_{kl} > 0$ and*

$d_i, d_j, d_k, d_l \geq 2$. *The edge clustering coefficients of* (p,q) *satisfy*

$$\xi_C(p,q) = \phi_{p,q}\,\xi_A(i,j)\,\xi_B(k,l),$$

where

$$\phi_{p,q} := \frac{(\min(d_i,d_j)-1)(\min(d_k,d_l)-1)}{\min(d_id_k, d_jd_l)-1},$$

which is in the interval $(0,1)$.

Proof. The derivation is merely plugging in the definition and reorganizing, similar to the proof of Thm. 1. To see that $\phi_{p,q}$ can be arbitrarily small, consider the case where $\min(d_i,d_j) = d_i$, $\min(d_k,d_l) = d_l$, yet $\min(d_id_k, d_jd_l) = d_id_k$, then

$$\phi_{p,q} = \frac{(d_i-1)(d_l-1)}{d_id_k-1},$$

which approaches zero as d_k and d_j get large. □

This shows if A and B have a relatively negative *assortativity* (more than expected high-degree vertices connected to low-degree vertices [20]), then there will be several edges (p,q) with much lower clustering coefficients than the product of clustering coefficients of the associated edges (i,j) and (k,l) from their factors. Although adding self loops can increase the clustering coefficients of edges, it cannot produce a controlled scaling law for all cases.

C. Probabilistic Edge Rejection

There are several potentially less-than-desirable features of non-stochastic Kronecker generators that can be mitigated in practice by probabilistic rejection of a small percentage of the edges, without losing the ability to reliably calculate the ground truth of local triangle statistics. These include unrealistic degree and triangle degree distribution: no large primes are possible; large holes in the distributions; excessive ties for large values in the distribution (as observed in [16] and [21]). More importantly, implicit or explicit use of the Kronecker formulas can make the computation of global triangle statistics sublinear in $|\mathcal{E}_C|$ (and local statistics linear), which is non-ideal for a good-faith benchmark. (For example, due to the Kronecker structure a spectral method can efficiently solve for large swathes of the eigenspace of C, which can be used to great advantage in some graph analytics without the algorithm developer even realizing it).

In this section, we describe a way to reliably generate large graphs with ground truth that are not exactly Kronecker graphs but have known local triangle statistics and improved degree/triangle distributions.

Def. 8. *Let* $hash(p,q)$ *be a specific hash function mapping* $\mathcal{E}_C \longrightarrow [0,1]$. *For* $\nu \in [0,1]$, *there exists a parameterized family* $\mathcal{G}_{C,\nu}$ *of subgraphs of* $\mathcal{G}_C$ *defined by:* $(p,q) \in \mathcal{G}_{C,\nu}$ *only if*

$$(p,q) \in \mathcal{G}_C \quad \textit{and} \quad hash(p,q) \leq \nu.$$

If we pick a few specific values of ν, say $\{1, 0.99, 0.95, 0.90\}$, then we are able to generate $\mathcal{G}_C, \mathcal{G}_{C,.99}, \mathcal{G}_{C,.95}, \mathcal{G}_{C,.9}$ jointly by storing the hash values of every edge. Then an algorithm that efficiently enumerates triangles in $\mathcal{G}_C$ (such as that from [22], [23]) can simultaneously enumerate triangles in $\mathcal{G}_{C,.99}, \mathcal{G}_{C,.95}, \mathcal{G}_{C,.9}$, as triangle (p_1,p_2,p_3) exists in $\mathcal{G}_{C,\nu}$ only if (p_1,p_2,p_3) exists in $\mathcal{G}_C$ and

$$\max\left(\text{hash}(p_1,p_2), \text{hash}(p_1,p_3), \text{hash}(p_2,p_3)\right) \leq \nu.$$

An individual triangle has probability ν^3 of existing in $\mathcal{G}_{C,\nu}$ and the expectation of vertex p's triangle count is $\nu^3 t_p$. The variance is more complicated due to dependence of edges in overlapping triangles. For edges $(p,q) \in \mathcal{G}_{C,\nu}$ the expectation of triangle count is $\nu^2\Delta_{p,q}$.

The ground truth of local triangle statistics of $\mathcal{G}_C$ are easily checked via Kronecker formulas, and the ground truth of $\mathcal{G}_{C,\nu}$ is computed by an algorithm that gets all local triangle statistics of $\mathcal{G}_C$ correct.

Clearly, bounds and expected values on the local triangle counts are possible for subgraphs of Kronecker graphs, and methods that use the counts for the Kronecker graph and delete edges are also possible. So this is not an approach that completely removes the possiblity of gaming the Kronecker structure in benchmarking scenarios. However, it makes it so accidental exploitation of the Kronecker structure is far less likely.

V. Distance-Based Centrality Metrics

Vertex centrality metrics are fairly ubiquitous graph-based features. Distance-based metrics (such as eccentricity, closeness centrality, and betweenness centrality) are typically quite expensive for large graphs. Direct calculation uses breadth-first search from every vertex and has worst-case bounds $O(|\mathcal{V}||\mathcal{E}|)$ [24], although several heuristic and/or approximation techniques exist for eccentricity [2] and closeness centrality [4]. Here, we demonstrate that Kronecker formulas exist that allow efficient ground truth computation of eccentricity at all vertices and closeness centrality at a subset of the vertices of Kronecker product graphs $\mathcal{G}_C$.

First we demonstrate Kronecker formulas for minimum hop length between two vertices and graph diameter, which we will leverage to derive the Kronecker formulas for the centrality metrics in the following sections.

Def. 9. *Let every vertex in* A *have a self loop, or* $A \circ I_A = I_A$. *The unweighted distance, or* hop count*, from* i *to* j *in* $\mathcal{G}_A$ *is*

$$hops_A(i,j) := \min_{h \in \mathbb{N}^+} \{\mathbf{e}_i^t A^h \mathbf{e}_j > 0\}.$$

We refer to the full set of distances, $hops_A(\cdot,\cdot)$, *as the* hop count matrix *and use the notation* $hops_A(i,\cdot)$ *to represent the row vector of all of vertex* i*'s hop counts.*

Notice $\mathbf{e}_i^t A^h \mathbf{e}_j > 0$ implies $\mathbf{e}_i^t A^{h'} \mathbf{e}_j > 0$ for any $h' > h$ when all vertices of A have self loops.

Thm. 3. *Let* $A \circ I_A = I_A$, $B \circ I_B = I_B$, *and* $C = A \otimes B$. *Then for* $(p,q) \in \mathcal{E}_C$, *the hop count from* p *to* q *in* $\mathcal{G}_C$ *satisfies*

$$hops_C(p,q) = \max\{hops_A(i,j), hops_B(k,l)\}$$

for $i = \alpha_{n_B}(p), j = \alpha_{n_B}(q), k = \beta_{n_B}(p)$, *and* $l = \beta_{n_B}(q)$.

Proof. $hops_C(p,q) =$

$$\begin{aligned} &= \min_{h\in\mathbb{N}^+}\{\mathbf{e}_p^t C^h \mathbf{e}_q > 0\} \\ &= \min_{h\in\mathbb{N}^+}\{(\mathbf{e}_i^t \otimes \mathbf{e}_k^t)(A^h \otimes B^h)(\mathbf{e}_j \otimes \mathbf{e}_l) > 0\} \\ &= \min_{h\in\mathbb{N}^+}\{(\mathbf{e}_i^t A^h \mathbf{e}_j)\cdot(\mathbf{e}_k^t B^h \mathbf{e}_l) > 0\} \\ &= \max\{\, hops_A(i,j)\,,\, hops_B(k,l)\,\}. \end{aligned}$$

□

Def. 10. *For a graph* A, *the* diameter *of* A *is defined by*

$$diam(\mathcal{G}_A) := \max_{(i,j)\in\mathcal{E}_A} hops_A(i,j)$$

Cor. 3. *Let* $A \circ I_A = I_A$, $B \circ I_B = I_B$, *and* $C = A \otimes B$. *Then*

$$diam(\mathcal{G}_C) = \max\{diam(\mathcal{G}_A), diam(\mathcal{G}_B)\}.$$

Proof. $diam(\mathcal{G}_C) =$

$$\begin{aligned} &= \max_{(p,q)\in\mathcal{E}_C} hops_C(p,q) \\ &= \max_{(i,j)\in\mathcal{E}_A}\max_{(k,l)\in\mathcal{E}_B} \max\{\, hops_A(i,j)\,,\, hops_B(k,l)\,\} \\ &= \max\{\, diam(\mathcal{G}_A)\,,\, diam(\mathcal{G}_B)\,\}. \end{aligned}$$

□

A. Vertex Eccentricity

Def. 11. *For a vertex* $i \in \mathcal{V}_A$, *the* eccentricity *of* i *is defined as*

$$\epsilon_A(i) := \max_{j\in\mathcal{V}_A} hops_A(i,j)$$

Cor. 4. *Let* $A \circ I_A = I_A$, $B \circ I_B = I_B$, *and* $C = A \otimes B$. *For any vertex* $p \in \mathcal{V}_C$,

$$\epsilon_C(p) = \max\{\epsilon_A(i), \epsilon_B(k)\}$$

for $i = \alpha_{n_B}(p)$ *and* $k = \beta_{n_B}(p)$.

Proof. $\epsilon_C(p) =$

$$\begin{aligned} &= \max_{q\in\mathcal{V}_C} hops_C(p,q) \\ &= \max_{j\in\mathcal{V}_A, l\in\mathcal{V}_B} \max\{\, hops_A(i,j)\,,\, hops_B(k,l)\,\} \\ &= \max\{\epsilon_A(i), \epsilon_B(k)\} \end{aligned}$$

□

We demonstrate this result by letting A represent a real-world scale-free graph and forming $C = A \otimes A$, and computing the vertex eccentricity of C using algorithms from [3]. We compare with those that come from applying Cor. 4 to the vertex eccentricities of A. Specifically, we chose A to be a `gnutella` graph (`gnutella08`) from the SNAP dataset [25]. The graph represents peer-to-peer file sharing activity. We formed the undirected version of the largest connected component, adding all self loops to A. Then we used our distributed HPC graph generator to form $C = A \otimes A$. We show full histograms of the vertex eccentricity in Fig. 1, and summarize the important graph properties below.

Data	Graph	Vertices	Edges
`gnutella08`	A	6.3K	21K
	$A \otimes A$	40M	1.1B

B. Closeness Centrality

Def. 12. *For a vertex* $i \in \mathcal{V}_A$, *the* closeness centrality *of* i *is defined by*

$$\zeta_A(i) := \sum_{j\in\mathcal{V}_A} \frac{1}{hops_A(i,j)}.$$

Thm. 4. *Let* $A \circ I_A = I_A$, $B \circ I_B = I_B$, *and* $C = A \otimes B$. *For any vertex* $p \in \mathcal{V}_C$,

$$\zeta_C(p) = \sum_{j\in\mathcal{V}_A}\sum_{l\in\mathcal{V}_B} \frac{1}{\max\{\, hops_A(i,j)\,,\, hops_B(k,l)\,\}}.$$

Proof. $\zeta_C(p) =$

$$\begin{aligned} &= \sum_{q\in\mathcal{V}_C} \frac{1}{hops_C(p,q)} \\ &= \sum_{j\in\mathcal{V}_A}\sum_{l\in\mathcal{V}_B} \frac{1}{\max\{\, hops_A(i,j)\,,\, hops_B(k,l)\,\}} \end{aligned}$$

□

To compute the closeness centrality at vertex $p \in \mathcal{V}_C$, we only need rows $hops_A(i,\cdot)$ and $hops_B(k,\cdot)$ of the hop count matrix. This shows with $\mathcal{O}(n_A + n_B)$ storage and $\mathcal{O}(n_A n_B)$ computation, we can compute $\zeta_C(p)$ as we build C. If we store $r << \min\{n_A, n_B\}$ columns of the hop count matrices for both A and B, this means we can compute r^2 values of ζ_C with similar order of memory, $\mathcal{O}(rn_A + rn_B)$ and cost $\mathcal{O}(r^2 n_A n_B)$.

However, we can take advantage of the low number of possibilities in $hops_A(i,\cdot)$ and $hops_B(k,\cdot)$ for small diameter $\mathcal{G}_A$ and $\mathcal{G}_B$ and reduce the $\mathcal{O}(n_A n_B)$ complexity of computing r^2 values of ζ_C. We sort the values in the columns $hops_A(i,\cdot)$ and $hops_B(k,\cdot)$ in decreasing order, and rewrite the summation to factor out equivalent values. Let $h^* = \max\{diam(\mathcal{G}_A), diam(\mathcal{G}_B)\}$.

$$\zeta_C(p) = \sum_{h=1}^{h^*} \frac{|\{(p,q)\in\mathcal{E}_C \,:\, hops_C(p,q) = h\}|}{h}.$$

Assuming $n_A > n_B$ (w.l.o.g.) we see that the cost is reduced to

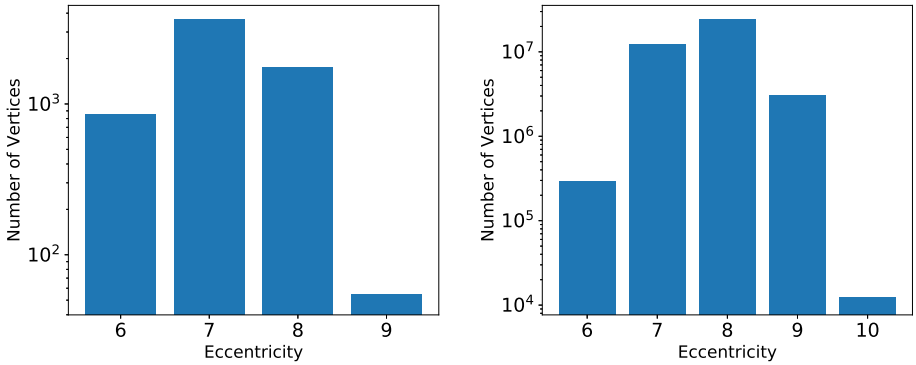
$$\mathcal{O}\Big(rn_A \log n_A + r^2 h^*\Big).$$

C. Controlling Diameter

The results in this section have assumed both A and B have self loops on all vertices. By similar arguments, we can loosen these requirements to get the following:

Thm. 5. *Let* $A \circ I_A = I_A$, $\mathcal{G}_B$ *be undirected, and* $C = A \otimes B$. *Then for* $(p,q) \in \mathcal{E}_C$, *the hop count from* p *to* q *in* $\mathcal{G}_C$ *satisfies*

$$\begin{aligned} \max\{hops_A(i,j), hops_B(k,l)\} &\le hops_C(p,q) \\ &\le \max\{hops_A(i,j), hops_B(k,l)\} + 1 \end{aligned}$$



(a) A (b) $C = A \otimes A$

Fig. 1: Experiment `gnutella` that demonstrates vertex eccentricities comply with a max-type scaling law, as implied by Cor. 4. We report an approximate vertex eccentricity distribution for C, as computed with algorithms from [3] (30% of vertices may be estimating a value 1 greater than actual eccentricity).

for $i = \alpha_{n_B}(p), j = \alpha_{n_B}(q), k = \beta_{n_B}(p)$, *and* $l = \beta_{n_B}(q)$.

and

Cor. 5. *Let* $A \circ I_A = I_A$, $\mathcal{G}_B$ *be undirected, and* $C = A \otimes B$. *Then*

$$\max\{diam(\mathcal{G}_A), diam(\mathcal{G}_B)\} \leq diam(\mathcal{G}_C) \leq \max\{diam(\mathcal{G}_A), diam(\mathcal{G}_B\} + 1.$$

Corollary 5 provides the ability to control the diameter of Kronecker graphs. By taking B to be any (possibly real-world) undirected graph and A to be a generated graph with self loops and a known large diameter, Kronecker graphs that incorporate the structure of B can be constructed to have large, controlled diameters. Similarly, we can choose A to have vertices with large eccentricities to produce a number of vertices in C with large eccentricities, giving more fine-grained control of the resulting graph.

VI. Community Structure

Traditionally, a graph with community structure has vertex sets with relatively high internal edge density and relatively low external edge density. In this section, we demonstrate that if factors A and B have strong community structure, then this feature is maintained by the Kronecker graph that has all self loops in both factors.

Def. 13. *Let* $\mathcal{S}_A \subset \mathcal{V}_A$ *and define* indicator vector $\mathbf{1}_{\mathcal{S}_A} \in \mathbb{B}^{n_A}$ *to have a 1 in the* i*-th entry if and only if* $i \in \mathcal{S}_A$. *The* internal edge count *is*

$$m_{in}(\mathcal{S}_A) := \frac{1}{2}\mathbf{1}_{\mathcal{S}_A}^t A \mathbf{1}_{\mathcal{S}_A},$$

whereas the external edge count *is*

$$m_{out}(\mathcal{S}_A) = \mathbf{1}_{\mathcal{S}_A}^t A \left(\mathbf{1}_A - \mathbf{1}_{\mathcal{S}_A}\right).$$

Internal and external edge densities *are*

$$\rho_{in}(\mathcal{S}_A) := \frac{2m_{in}(\mathcal{S}_A)}{|\mathcal{S}_A|(|\mathcal{S}_A| - 1)}, \quad \rho_{out} := \frac{m_{out}(\mathcal{S}_A)}{|\mathcal{S}_A|(n_A - |\mathcal{S}_A|)}.$$

Def. 14. *The* Kronecker product of vertex sets $\mathcal{S}_A \in \mathcal{V}_A$ *and* $\mathcal{S}_B \in \mathcal{V}_B$ *is*

$$\mathcal{S}_C = \mathcal{S}_A \otimes \mathcal{S}_B := supp(\mathbf{1}_A \otimes \mathbf{1}_B)$$

Thm. 6. (Internal/External Edge Counts) *Let* $A \circ I_A = O_A$, $B \circ I_B = O_B$. *Let both factors have self loops for all vertices added in* $C = (A + I_A) \otimes (B + I_B)$. *Let* $\mathcal{S}_C = \mathcal{S}_A \otimes \mathcal{S}_B$ *Then,* $m_{in}(\mathcal{S}_C) =$

$$2m_{in}(\mathcal{S}_A)m_{in}(\mathcal{S}_B) + m_{in}(\mathcal{S}_A)|\mathcal{S}_B| + |\mathcal{S}_A|m_{in}(\mathcal{S}_B),$$

and $m_{out}(\mathcal{S}_C) =$

$$m_{out}(\mathcal{S}_A)\left[\frac{1}{2}m_{out}(\mathcal{S}_B) + |\mathcal{S}_B| + 2m_{in}(\mathcal{S}_B)\right] + m_{out}(\mathcal{S}_B)\left[\frac{1}{2}m_{out}(\mathcal{S}_A) + |\mathcal{S}_A| + 2m_{in}(\mathcal{S}_A)\right].$$

The proof of the previous result is in the appendix. We use Thm. 6 to show that $\mathcal{G}_C$ has sets whose internal density follows a controlled scaling law (bounded from below).

Cor. 6. *If* $|\mathcal{S}_A|, |\mathcal{S}_B| > 1$, *then*

$$\rho_{in}(\mathcal{S}_C) \geq \frac{1}{3}\rho_{in}(\mathcal{S}_A)\rho_{in}(\mathcal{S}_B)$$

Proof.

$$\rho_{in}(\mathcal{S}_C) = \frac{2m_{in}(\mathcal{S}_C)}{|\mathcal{S}_C|(|\mathcal{S}_C|-1)} = \frac{2(2m_{in}(\mathcal{S}_A)m_{in}(\mathcal{S}_B)+m_{in}(\mathcal{S}_A)|\mathcal{S}_B|+|\mathcal{S}_A|m_{in}(\mathcal{S}_B))}{|\mathcal{S}_A||\mathcal{S}_B|(|\mathcal{S}_A||\mathcal{S}_B|-1)} \geq \frac{2(2m_{in}(\mathcal{S}_A)m_{in}(\mathcal{S}_B))}{|\mathcal{S}_A||\mathcal{S}_B|(|\mathcal{S}_A||\mathcal{S}_B|-1)} = \theta_{|\mathcal{S}_A|,|\mathcal{S}_B|}\rho_{in}(\mathcal{S}_A)\rho_{in}(\mathcal{S}_B),$$

where

$$\theta_{|\mathcal{S}_A|,|\mathcal{S}_B|} := \frac{(|\mathcal{S}_A|-1)(|\mathcal{S}_B|-1)}{|\mathcal{S}_A||\mathcal{S}_B|-1}.$$

which is greater than $1/3$, as in Thm 1.

□

Additionally, given the assumption that there are significantly many external edges (often true for communities where $|\mathcal{S}_A| << n_A$), a simple bound can be derived to show external densities also follow a controlled scaling law (bounded from above).

Cor. 7. *Let $m_{out}(\mathcal{S}_A) \geq |\mathcal{S}_A|$, $m_{out}(\mathcal{S}_B) \geq |\mathcal{S}_B|$, and*

$$\omega := \max\left(\frac{m_{in}(\mathcal{S}_A)}{m_{out}(\mathcal{S}_A)}, \frac{m_{in}(\mathcal{S}_B)}{m_{out}(\mathcal{S}_B)}\right).$$

Then we have the following scaling law

$$\rho_{out}(\mathcal{S}_C) \leq (1+3\omega)\Omega_{|\mathcal{S}_A|,|\mathcal{S}_B|}\rho_{out}(\mathcal{S}_A)\rho_{out}(\mathcal{S}_B),$$

where

$$\Omega_{|\mathcal{S}_A|,|\mathcal{S}_B|} := \frac{1+|\mathcal{S}_A||\mathcal{S}_B|n_A^{-1}n_B^{-1}}{1-|\mathcal{S}_A||\mathcal{S}_B|n_A^{-1}n_B^{-1}}$$

is a number slightly greater than 1 when $|\mathcal{S}_A| << n_A$ and $|\mathcal{S}_B| << n_B$.

Proof. By assumption

$$m_{out}(\mathcal{S}_C) \leq (1+3\omega)m_{out}(\mathcal{S}_A)m_{out}(\mathcal{S}_B),$$

which yields

$$\begin{aligned}\rho_{out}(\mathcal{S}_C) &= \frac{m_{out}(\mathcal{S}_C)}{|\mathcal{S}_C|(n_C-|\mathcal{S}_C|)} \\ &\leq \frac{(1+3\omega)m_{out}(\mathcal{S}_A)m_{out}(\mathcal{S}_B)}{|\mathcal{S}_A||\mathcal{S}_B|(n_An_B-|\mathcal{S}_A||\mathcal{S}_B|)} \\ &\leq (1+3\omega)\Omega_{|\mathcal{S}_A|,|\mathcal{S}_B|}\rho_{out}(\mathcal{S}_A)\rho_{out}(\mathcal{S}_B).\end{aligned}$$

□

Def. 15. (Non-overlapping vertex partition) *Let Π_A be a* non-overlapping vertex partition *of $\mathcal{V}_A$, defined by a_{max} vertex sets*

$$\Pi_A := \left\{\mathcal{S}_A^{(a)}\right\}_{a=1}^{a_{max}},$$

that satisfy $\mathcal{S}_A^{(a)} \cap \mathcal{S}_A^{(a')} = \emptyset$, for $a \neq a'$ and

$$\bigcup_{a=1}^{a_{max}} \mathcal{S}_A^{(a)} = \mathcal{V}_A.$$

Def. 16. (Kronecker Partition) *Given partitions Π_A of $\mathcal{V}_A$ and Π_B of $\mathcal{V}_B$, the* Kronecker partition $\Pi_C = \Pi_A \otimes \Pi_B$ *is the $c_{max} := a_{max}b_{max}$ non-overlapping vertex sets defined by*

$$\Pi_C := \left\{\mathcal{S}_A^{(a)} \otimes \mathcal{S}_B^{(b)}\right\}$$

for $a = 1, ..., a_{max}$ and $b = 1, ..., b_{max}$.

Ex. 1. *A simple example is seen by letting $\mathcal{G}_A$ be x_A disjoint cliques of size y_A and $\mathcal{G}_B$ be x_B disjoint cliques of size y_B. Then, $C = (A+I_A)\otimes(B+I_B)$ will be x_Ax_B disjoint cliques of size y_Ay_B.*

Slightly more generally, let factors A and B be stochastic block models with x_A and x_B blocks, and with internal edge densities ρ_0 and external edge densities ρ_1. If the graph factors are of significant size, then, $C = (A+I_A)\otimes(B+I_B)$ will have $\rho_{in}(\mathcal{S}_C) \approx \rho_0^2$ and $\rho_{out}(\mathcal{S}_C) \approx \rho_1^2$.

A. Community Density Experiment

We demonstrate the scaling laws in Cor. 6 and Cor. 7 for a small $\mathcal{G}_A$ with strong community structure. We let A be the graph `groundtruth_20000` from GraphChallenge dataset [14], and set $C = (A+I_A)\otimes(A+I_A)$. The 33 ground truth communities were also mapped onto 1089 communities in $\mathcal{V}_C$ using Kronecker products, as in Def 16. We plot the internal edge density versus external edge density in Figure 2, where the scaling laws are validated.

	A	$C = (A+I_A)\otimes(A+I_A)$
$\mathcal{V}_A$	20,000	400,000,000
$\mathcal{E}_A$	408,778	83,549,726,642
# comms	33	1089
ρ_{in}	[3e-2, 1e-1]	[1e-3, 1.2e-2]
ρ_{out}	[2.5e-4, 5.5e-4]	[5e-7, 3e-6]

VII. Conclusion

This work presented an open-source distributed Kronecker generator that produces the product $C = A \otimes B$ given files containing A and B. It also provided many uses of nonstochastic Kronecker products to efficiently calculate or control several graph analytics on a large Kronecker graph using the analytics on the smaller factors. The analytics considered here include triangle counts at vertices and edges, clustering coefficients, various distance metrics, and internal/external community edge counts and densities. These results provide an additional tool for validation of results at large scales. They also present the possibility of use in benchmarks, and while the Kronecker structure can still be intentionally exploited by algorithms, there are ways to avoid accidental use of this structure for triangle counting. Further work may provide more uses for nonstochastic Kronecker graphs in validation and benchmarking of algorithms.

Appendix

A. Proofs of Selected Results

Proof of Corr. 1.

Proof. $(A+I_A)^3 \otimes (B+I_B)^3 =$

$$\begin{array}{cccc} A^3\otimes B^3 & +3A^3\otimes B^2 & +3A^3\otimes B & +A^3\otimes I_B \\ +3A^2\otimes B^3 & +9A^2\otimes B^2 & +9A^2\otimes B & +3A^2\otimes I_B \\ +3A\otimes B^3 & +9A\otimes B^2 & +9A\otimes B & +3A\otimes I_B \\ +I_A\otimes B^3 & +3I_A\otimes B^2 & +3I_A\otimes B & +I_A\otimes I_B. \end{array}$$

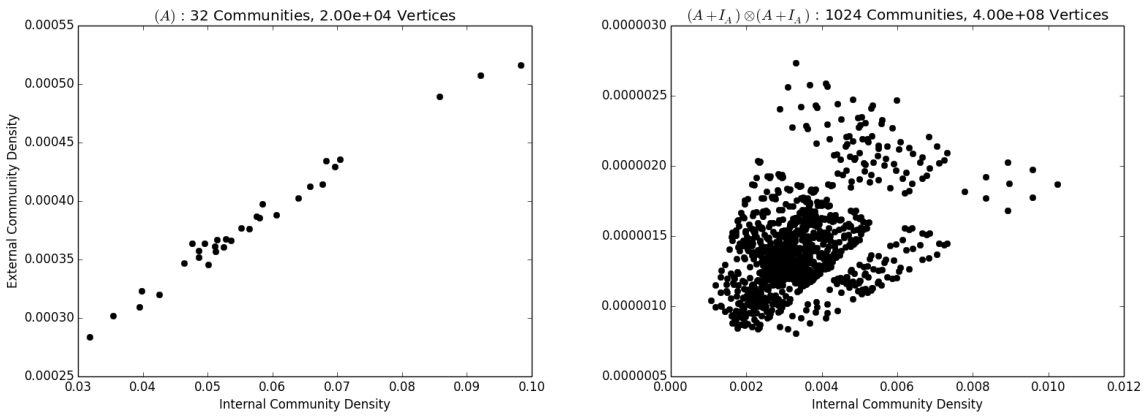


Fig. 2: Experiment `groundtruth_20000` that demonstrates relatively high internal edge density and relatively low external edge density are guaranteed when factor matrices have communities, as implied by Cor. 6 and Cor. 7.

$-3(A+I_A)^2 \otimes (B+I_B)^2 =$

$$\begin{array}{ccc} -3A^2\otimes B^2 & -6A^2\otimes B & -3A^2\otimes I_B \\ -6A\otimes B^2 & -12A\otimes B & -6A\otimes I_B \\ -3I_A\otimes B^2 & -6I_A\otimes B & -3I_A\otimes I_B. \end{array}$$

$+3(A+I_A)\otimes(B+I_B) =$

$$\begin{array}{cc} 3A\otimes B & +3A\otimes I_B \\ +3I_A\otimes B & +3I_A\otimes I_B. \end{array}$$

$(C-I_C)^3 = C^3 - 3C^2 + 3C - I_C =$

$$\begin{array}{cccc} A^3\otimes B^3 & +3A^3\otimes B^2 & +3A^3\otimes B & +A^3\otimes I_B \\ +3A^2\otimes B^3 & +6A^2\otimes B^2 & +3A^2\otimes B & \\ +3A\otimes B^3 & +3A\otimes B^2 & & \\ +I_A\otimes B^3. & & & \end{array}$$

$\mathbf{e}_i^t A\mathbf{e}_i = \mathbf{e}_k^t B\mathbf{e}_k = 0$, thus

$$\begin{aligned} t_p &= \tfrac{1}{2}\mathbf{e}_p^t(C-I)^3\mathbf{e}_p \\ &= 2t_it_k + 3(t_id_k + d_id_k + d_it_k) + t_i + t_k. \end{aligned}$$

□

Proof of Corr. 2.

Proof. The edge participation is $(C-I)^2\circ(C-I) = (C^2 - 2C+I)\circ(C-I)$. We break it up and combine the terms after simplifying, $C^2\circ(C-I) =$

$$\left\{\begin{array}{ccc} A^2\otimes B^2 & +2A^2\otimes B & +A^2\otimes I_B \\ +2A\otimes B^2 & +4A\otimes B & +2A\otimes I_B \\ I_A\otimes B^2 & +2I_A\otimes B & +I_A\otimes I_B \end{array}\right\}\circ\left\{\begin{array}{c} A\otimes B \\ +A\otimes I_B \\ +I_A\otimes B \end{array}\right\}$$

Noting $I_A\circ A = O_A$, $A\circ A = A$, $I_A\circ I_A = I_A$ (and similarly for B), we see $C^2\circ(C-I) =$

$$\begin{array}{lcl} \left[(A^2\circ A) + 2(A) + (A^2\circ I_A) + (I_A)\right] & \otimes & (B^2\circ B) \\ +\left[2(A^2\circ A) + 4(A) + 2(A^2\circ I_A) + 2(I_A)\right] & \otimes & (B) \\ +\left[(A^2\circ A) + 2(A)\right] & \otimes & (B^2\circ I_B) \\ +\left[(A^2\circ A) + 2(A)\right] & \otimes & (I_B). \end{array}$$

Then, $C\circ C = C$, $C\circ I_C = I_C$, so $(-2C+I)\circ(C-I) =$

$$\begin{aligned} &= -2C + 2I_C \\ &= -2A\otimes B - 2A\otimes I_B - 2I_A\otimes B - I_A\otimes I_B. \end{aligned}$$

Plugging in $\mathbf{e}_p = (\mathbf{e}_i\otimes\mathbf{e}_j)$ and $\mathbf{e}_q = (\mathbf{e}_k\otimes\mathbf{e}_l)$, yields $\mathbf{e}_p^t[(C^2-2C+I)\circ(C-I)]\mathbf{e}_q =$

$$\begin{array}{c} \Delta_{ij}\Delta_{kl} + 2(\Delta_{ij}+\Delta_{kl}) \\ +\Delta_{ij}(d_k+1)\delta(k,l) + \Delta_{kl}(d_i+1)\delta(i,j) \\ +2(d_i\delta(i,j) + d_k\delta(k,l) + 1) + \delta(i,j)\delta(k,l). \end{array}$$

because $\mathbf{e}_i^t(A^2\circ I_A)\mathbf{e}_j = \delta(i,j)$, $\mathbf{e}_i^t(A^2\circ I_A)\mathbf{e}_j = d_i\delta(i,j)$, with similar equalities for B. Note that $p\neq q$ means that both $i=j$ and $k=l$ cannot hold, and we see the result.

□

Proof of Thm. 6.

Proof. $m_{in}(\mathcal{S}_C) = \frac{1}{2}\mathbf{1}_{\mathcal{S}_C}^t\left[C-I_C\right]\mathbf{1}_{\mathcal{S}_C} =$

$$\begin{array}{c} \frac{1}{2}\mathbf{1}_{\mathcal{S}_C}^t\left[(A\otimes B) + (A\otimes I_B) + (I_A\otimes B)\right]\mathbf{1}_{\mathcal{S}_C} \\ = \frac{1}{2}\left(\mathbf{1}_{\mathcal{S}_A}^t A\mathbf{1}_{\mathcal{S}_A}\right)\left(\mathbf{1}_{\mathcal{S}_B}^t B\mathbf{1}_{\mathcal{S}_B}\right) + \\ \frac{1}{2}\left(\mathbf{1}_{\mathcal{S}_A}^t A\mathbf{1}_{\mathcal{S}_A}\right)\left(\mathbf{1}_{\mathcal{S}_B}^t\mathbf{1}_{\mathcal{S}_B}\right) + \\ \frac{1}{2}\left(\mathbf{1}_{\mathcal{S}_A}^t\mathbf{1}_{\mathcal{S}_A}\right)\left(\mathbf{1}_{\mathcal{S}_B}^t B\mathbf{1}_{\mathcal{S}_B}\right) \\ = 2m_{in}(\mathcal{S}_A)m_{in}(\mathcal{S}_B) + m_{in}(\mathcal{S}_A)|\mathcal{S}_B| + |\mathcal{S}_A|m_{in}(\mathcal{S}_B). \end{array}$$

For $m_{in}(\mathcal{S}_C)$, first note that $\mathbf{1}_{\mathcal{S}_A}^t A\mathbf{1}_A = 2m_{in}(\mathcal{S}_A) + m_{out}(\mathcal{S}_A)$ (and likewise for $\mathcal{S}_B$). Then $\mathbf{1}_{\mathcal{S}_C}^t\left[C-I_C\right]\mathbf{1}_C =$

$$\begin{array}{c} \mathbf{1}_{\mathcal{S}_C}^t\left[(A\otimes B) + (A\otimes I_B) + (I_A\otimes B)\right]\mathbf{1}_C \\ = \left(\mathbf{1}_{\mathcal{S}_A}^t A\mathbf{1}_A\right)\left(\mathbf{1}_{\mathcal{S}_B}^t B\mathbf{1}_B\right) + \left(\mathbf{1}_{\mathcal{S}_A}^t A\mathbf{1}_A\right)\left(\mathbf{1}_{\mathcal{S}_B}^t\mathbf{1}_B\right) \\ + \left(\mathbf{1}_{\mathcal{S}_A}^t\mathbf{1}_A\right)\left(\mathbf{1}_{\mathcal{S}_B}^t B\mathbf{1}_B\right) \\ = \left(2m_{in}(\mathcal{S}_A) + m_{out}(\mathcal{S}_A)\right)\left(2m_{in}(\mathcal{S}_B) + m_{out}(\mathcal{S}_B)\right) \\ +|\mathcal{S}_A|\left(2m_{in}(\mathcal{S}_B) + m_{out}(\mathcal{S}_B)\right) \\ +|\mathcal{S}_B|\left(2m_{in}(\mathcal{S}_A) + m_{out}(\mathcal{S}_A)\right) \end{array}$$

Lastly, $m_{out}(\mathcal{S}_C) = \mathbf{1}_{\mathcal{S}_C}^t [C - I_C] \left(\mathbf{1}_C - \mathbf{1}_{\mathcal{S}_C}^t\right) =$

$$\mathbf{1}_{\mathcal{S}_C}^t [C - I_C] \mathbf{1}_C - 2m_{in}(\mathcal{S}_C) =$$
$$m_{out}(\mathcal{S}_A)m_{out}(\mathcal{S}_B) + m_{out}(\mathcal{S}_A)\left(|\mathcal{S}_B| + 2m_{out}(\mathcal{S}_B)\right) +$$
$$m_{out}(\mathcal{S}_B)\left(|\mathcal{S}_A| + 2m_{out}(\mathcal{S}_A)\right).$$

□

Proof. $m_{in}(\mathcal{S}_C) = \frac{1}{2}\mathbf{1}_{\mathcal{S}_C}^t [C - I_C] \mathbf{1}_{\mathcal{S}_C} =$

$$\frac{1}{2}\mathbf{1}_{\mathcal{S}_C}^t [(A \otimes B) + (A \otimes I_B) + (I_A \otimes B)] \mathbf{1}_{\mathcal{S}_C}$$
$$= \frac{1}{2}\left(\mathbf{1}_{\mathcal{S}_A}^t A \mathbf{1}_{\mathcal{S}_A}\right)\left(\mathbf{1}_{\mathcal{S}_B}^t B \mathbf{1}_{\mathcal{S}_B}\right) +$$
$$\frac{1}{2}\left(\mathbf{1}_{\mathcal{S}_A}^t A \mathbf{1}_{\mathcal{S}_A}\right)\left(\mathbf{1}_{\mathcal{S}_B}^t \mathbf{1}_{\mathcal{S}_B}\right) +$$
$$\frac{1}{2}\left(\mathbf{1}_{\mathcal{S}_A}^t \mathbf{1}_{\mathcal{S}_A}\right)\left(\mathbf{1}_{\mathcal{S}_B}^t B \mathbf{1}_{\mathcal{S}_B}\right)$$
$$= 2m_{in}(\mathcal{S}_A)m_{in}(\mathcal{S}_B) + m_{in}(\mathcal{S}_A)|\mathcal{S}_B| + |\mathcal{S}_A|m_{in}(\mathcal{S}_B).$$

For $m_{in}(\mathcal{S}_C)$, first note that $\mathbf{1}_{\mathcal{S}_A}^t A \mathbf{1}_A = 2m_{in}(\mathcal{S}_A) + m_{out}(\mathcal{S}_A)$ (and likewise for $\mathcal{S}_B$). Then $\mathbf{1}_{\mathcal{S}_C}^t [C - I_C] \mathbf{1}_C =$

$$\mathbf{1}_{\mathcal{S}_C}^t [(A \otimes B) + (A \otimes I_B) + (I_A \otimes B)] \mathbf{1}_C$$
$$= \left(\mathbf{1}_{\mathcal{S}_A}^t A \mathbf{1}_A\right)\left(\mathbf{1}_{\mathcal{S}_B}^t B \mathbf{1}_B\right) + \left(\mathbf{1}_{\mathcal{S}_A}^t A \mathbf{1}_A\right)\left(\mathbf{1}_{\mathcal{S}_B}^t \mathbf{1}_B\right)$$
$$+ \left(\mathbf{1}_{\mathcal{S}_A}^t \mathbf{1}_A\right)\left(\mathbf{1}_{\mathcal{S}_B}^t B \mathbf{1}_B\right)$$
$$= \left(2m_{in}(\mathcal{S}_A) + m_{out}(\mathcal{S}_A)\right)\left(2m_{in}(\mathcal{S}_B) + m_{out}(\mathcal{S}_B)\right)$$
$$+|\mathcal{S}_A|\left(2m_{in}(\mathcal{S}_B) + m_{out}(\mathcal{S}_B)\right)$$
$$+|\mathcal{S}_B|\left(2m_{in}(\mathcal{S}_A) + m_{out}(\mathcal{S}_A)\right)$$

Lastly, $m_{out}(\mathcal{S}_C) = \mathbf{1}_{\mathcal{S}_C}^t [C - I_C] \left(\mathbf{1}_C - \mathbf{1}_{\mathcal{S}_C}^t\right) =$

$$\mathbf{1}_{\mathcal{S}_C}^t [C - I_C] \mathbf{1}_C - 2m_{in}(\mathcal{S}_C) =$$
$$m_{out}(\mathcal{S}_A)m_{out}(\mathcal{S}_B) + m_{out}(\mathcal{S}_A)\left(|\mathcal{S}_B| + 2m_{out}(\mathcal{S}_B)\right) +$$
$$m_{out}(\mathcal{S}_B)\left(|\mathcal{S}_A| + 2m_{out}(\mathcal{S}_A)\right).$$

□

REFERENCES

[1] P. M. Weichsel, "The Kronecker product of graphs," *Proceedings of the American Mathematical Society*, vol. 13, no. 1, pp. 47–52, 1962.

[2] A. Backurs, L. Roditty, G. Segal, V. V. Williams, and N. Wein, "Towards tight approximation bounds for graph diameter and eccentricities," *CoRR*, vol. abs/1808.08494, 2018. [Online]. Available: http://arxiv.org/abs/1808.08494

[3] K. H. K. Iwabuchi, G. Sanders and R. Pearce, "Computing exact vertex eccentricity on massive-scale distributed graphs," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2018.

[4] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck, "Computing classic closeness centrality, at scale," in *Proceedings of the Second ACM Conference on Online Social Networks*, ser. COSN '14. New York, NY, USA: ACM, 2014, pp. 37–50. [Online]. Available: http://doi.acm.org/10.1145/2660460.2660465

[5] T. Reza, C. Klymko, M. Ripeanu, G. Sanders, and R. Pearce, "Towards practical and robust labeled pattern matching in trillion-edge graphs," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2017, pp. 1–12.

[6] M. Halappanavar, H. Lu, A. Kalyanaraman, and A. Tumeo, "Scalable static and dynamic community detection using grappolo," in *High Performance Extreme Computing Conference (HPEC)*. IEEE, 2017.

[7] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li, *RolX: Structural role extraction and mining in large graphs*, 2012, pp. 1231–1239.

[8] J. Kepner, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, T. Davis, V. Gadepally, M. Houle, M. Hubbell, H. Jananthan, M. Jones, A. Klein, P. Michaleas, R. Pearce, L. Milechin, J. Mullen, A. Prout, A. Rosa, G. Sanders, C. Yee, and A. Reuther, "Design, generation, and validation of extreme scale power-law graphs," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2018, pp. 279–286.

[9] C. F. Van Loan, "The ubiquitous Kronecker product," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1, pp. 85–100, 2000.

[10] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. New York, NY, USA: Cambridge University Press, 2012.

[11] G. Sanders, R. Pearce, T. L. Fond, and J. Kepner, "On large-scale graph generation with validation of diverse triangle statistics at edges and vertices," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2018, pp. 287–296.

[12] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A recursive model for graph mining," in *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM, 2004, pp. 442–446.

[13] J. Ang, B. Barrett, K. Wheeler, and R. Murphy, "Introducing the graph 500," 01 2010. [Online]. Available: https://graph500.org/

[14] S. Samsi, V. Gadepally, M. Hurley, M. Jones, E. Kao, S. Mohindra, P. Monticciolo, A. Reuther, S. Smith, W. Song, D. Staheli, and J. Kepner, "Static graph challenge: Subgraph isomorphism," in *High Performance Extreme Computing Conference (HPEC)*. IEEE, 2017.

[15] E. Kao, V. Gadepally, M. Hurley, M. Jones, J. Kepner, S. Mohindra, P. Monticciolo, A. Reuther, S. Samsi, W. Song, D. Staheli, and S. Smith, "Streaming Graph Challenge - Stochastic Block Partition," in *High Performance Extreme Computing Conference (HPEC)*. IEEE, 2017.

[16] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *J. Mach. Learn. Res.*, vol. 11, pp. 985–1042, mar 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1756039

[17] J. Kepner and J. Gilbert, *Graph algorithms in the language of linear algebra*. SIAM, 2011.

[18] "Havoqgt." [Online]. Available: https://github.com/LLNL/havoqgt

[19] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, 1998.

[20] M. E. Newman, "Assortive mixing in networks," *prl*, vol. 89, Oct 2002.

[21] B. Priest, "Sublinear approximation of centrality indices in large graphs," Ph.D. dissertation, Dartmouth, 2019.

[22] N. Chiba and T. Nishizeki, "Arboricity and subgraph listing algorithms," *SIAM Journal on Computing*, vol. 14, no. 1, pp. 210–223, 1985. [Online]. Available: https://doi.org/10.1137/0214017

[23] R. Pearce, "Triangle counting for scale-free graphs at scale in distributed memory," in *High Performance Extreme Computing Conference (HPEC)*. IEEE, 2017.

[24] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[25] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, jun 2014.