# Online Factorization and Partition of Complex Networks by Random Walk


Lin F. Yang*, Zheng Yu*, Vladimir Braverman°, Tuo Zhao◇, Mengdi Wang*

*Princeton University °Johns Hopkins University ◇Georgia Tech


## Background

- Large complex Networks are hard to understand and often not fully/explicitly observable

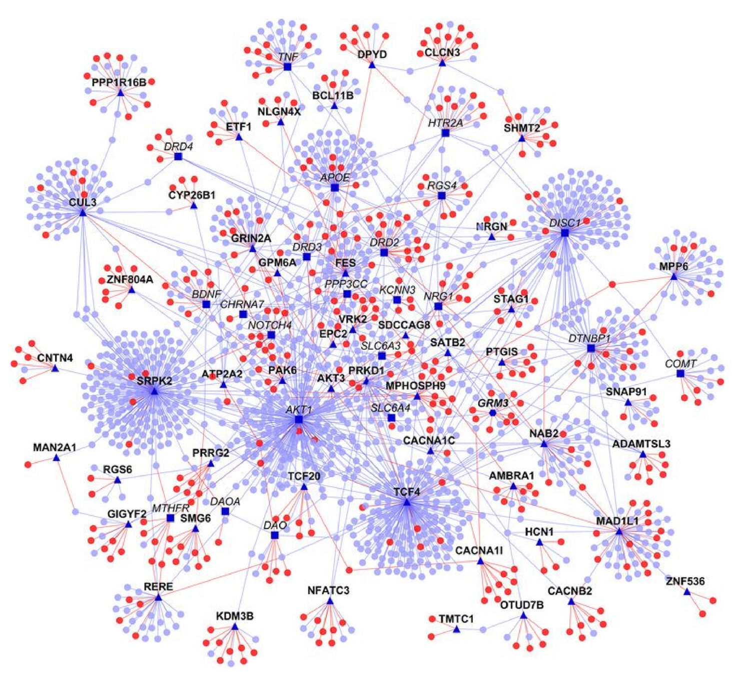
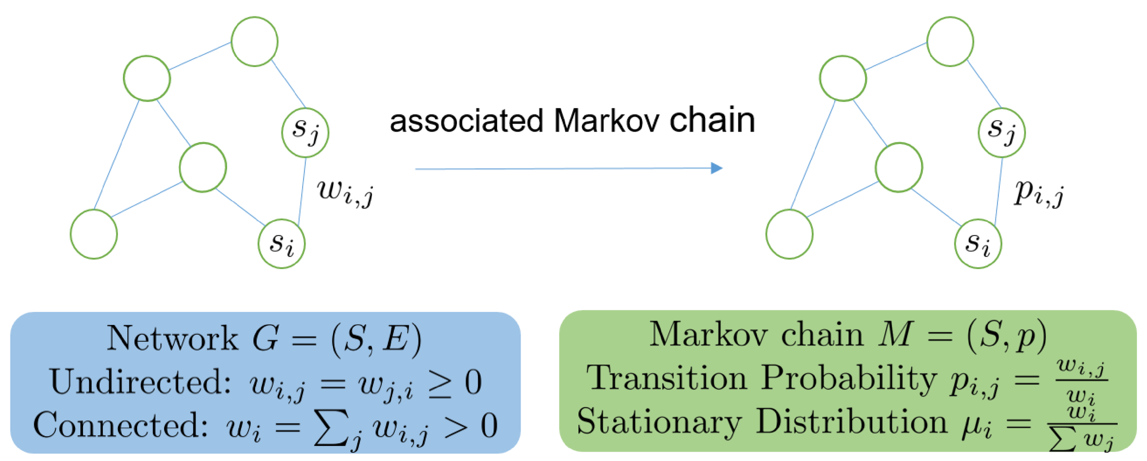


Figure 1: Protein-Protein Interaction Network

- Learning from a "random walk" associated with the network




**Problem of Interest:** Given streaming sample trajectory of state transition of the unknown Markov chain:
- how to extract its reduced-order information?
- how to recover the latent network partition?

in an **online** fashion with **low** computational and storage cost, and **small** sample complexity.

## Model Assumptions

**Nearly Low-Rankness:** Denote $\boldsymbol{D} = \mathrm{diag}\{\mu_i\}$ and $\boldsymbol{P} = (p_{i,j})$ as stationary distribution and transition matrix, the product $\boldsymbol{DP}$ has a strictly positive singular gap, i.e., it has the decomposition:

$$\boldsymbol{DP} = \boldsymbol{U}\mathrm{diag}(\sigma_1, \cdots, \sigma_r)\boldsymbol{V}^\top + \boldsymbol{F} \text{ and } \sigma_r > \|\boldsymbol{F}\|_2.$$

**Meta-State Structure:** The associated Markov chain has a lumpable structure, i.e., there exists a state partition $S = S_1 \cup S_2 \ldots \cup S_r$, such that

$$\forall\ s_k, s_h \in S_i,\ \forall\ j \in \{1, \cdots, r\},\ \sum_{s_\ell \in S_j} p_{k,\ell} = \sum_{s_\ell \in S_j} p_{h,\ell}$$

**Remark** (Weinan et al., 2008). *If Markov chain is lumpable and low rank, then one can recover its exact partition by clustering the reduced-order representations $\boldsymbol{M} = \boldsymbol{D}^{-1}\boldsymbol{V}$.*

## Learning Reduced-Order State Representation

**Down sampling to handle data dependency:**

$s^{(1)}, s^{(2)}, \ldots, s^{(\tau)}$, $s^{(\tau+1)}, \ldots, s^{(2\tau)}$, $\ldots$, $s^{((b-1)\tau+1)}, \ldots, s^{(b\tau)}$

Construct sample matrix $\boldsymbol{Z}^{(k)} = \mathbb{1}\{s^{(k\tau-1)}, s^{(k\tau)}\}$
- $\boldsymbol{Z}^{(k)}$'s are close to i.i.d. samples and
- $\mathbb{E}\left[\boldsymbol{Z}^{(k)} \middle| s^{(0)}\right] \approx \boldsymbol{DP} = \boldsymbol{U}\mathrm{diag}(\sigma_1, \cdots, \sigma_r)\boldsymbol{V}^\top + \boldsymbol{F}$

**Estimating $\boldsymbol{V}$ reduces back to streaming PCA problem!**

$$(\boldsymbol{U}^*, \boldsymbol{V}^*) = \operatorname*{arg\,max}_{\widetilde{\boldsymbol{U}}, \widetilde{\boldsymbol{V}} \in \mathbb{R}^{m\times r}} \mathrm{tr}\left[\widetilde{\boldsymbol{U}}^\top \mathbb{E}\boldsymbol{Z}\widetilde{\boldsymbol{V}}\right]$$

$$\text{subject to } \widetilde{\boldsymbol{U}}^\top\widetilde{\boldsymbol{U}} = \widetilde{\boldsymbol{V}}^\top\widetilde{\boldsymbol{V}} = \boldsymbol{I}_r.$$

Recast into a symmetric problem:

$$\boldsymbol{W}^* = \operatorname*{argmax}_{\boldsymbol{W} \in \mathbb{R}^{2m\times r}} \mathrm{tr}\left[\boldsymbol{W}^\top \mathbb{E}\boldsymbol{A}\boldsymbol{W}\right] \quad \text{subject to} \quad \boldsymbol{W}^\top\boldsymbol{W} = I_r,$$

where $\mathbb{E}\boldsymbol{A} = \begin{bmatrix} 0_{m\times m} & \mathbb{E}\boldsymbol{Z} \\ \mathbb{E}\boldsymbol{Z}^\top & 0_{m\times m} \end{bmatrix}$ and $\boldsymbol{W} = \frac{1}{\sqrt{2}}\left[\boldsymbol{U}^\top, \boldsymbol{V}^\top\right]^\top$.

**Generalized Hebbian Algorithm (GHA)**

**Input**: sample trajectory $s^{(k)}$, block size $\tau$, step size $\eta$.
**Init**: set random $\boldsymbol{W}^{(0)}$ with orthonormal columns;
set $k \leftarrow 1$.
**Repeat**:
Construct matrix $\boldsymbol{A}^{(k)}$ from sample $s^{(k\tau-1)}, s^{(k\tau)}$;
$\boldsymbol{W}^{(k+1)} \leftarrow \boldsymbol{W}^{(k)} + \eta(\boldsymbol{A}^{(k)}\boldsymbol{W}^{(k)} - \boldsymbol{W}^{(k)}\boldsymbol{W}^{(k)\top}\boldsymbol{A}^{(k)}\boldsymbol{W}^{(k)})$
$s \leftarrow s + 1$;
**Until** stopping condition is satisfied
**Output:** $[\widehat{U}; \widehat{V}] \leftarrow \sqrt{2}\boldsymbol{W}^{(k)}$.

★ **Streaming** algorithm with **low** computational cost $\mathcal{O}(|S|r)$ and **low** storage cost $\mathcal{O}(|S|r)$!

## Recovering Network Partition

**(1)** Run GHA on Markov transition data to obtain $[\widehat{\boldsymbol{U}}; \widehat{\boldsymbol{V}}]$.
**(2)** Let $\widehat{\boldsymbol{\mu}}$ be the empirical estimate of the stationary distribution, i.e.,

$$\widehat{\mu}_i = \sum_{k=1}^{n} \mathbb{I}(s^{(k)} = i)/n.$$

Let $\widehat{\boldsymbol{D}} = \mathrm{diag}(\widehat{\mu}_1, \widehat{\mu}_2, \ldots, \widehat{\mu}_m)$. Each row of $\widehat{\boldsymbol{M}} = \widehat{\boldsymbol{D}}^{-1}\widehat{\boldsymbol{V}}$ gives an approximate $r$-dimensional representation for the corresponding state/vertex.
**(3)** Find a set of centers $C = \{c_1, c_2, \ldots, c_r\} \subset \mathbb{R}^r$ by solving the following problem:

$$\widehat{C} = \mathrm{argmin}_C \textstyle\sum_{i=1}^m \min_{c \in C} d^2(\widehat{\boldsymbol{M}}_{i*}, c),$$

where $d(\widehat{\boldsymbol{M}}_{s_i*}, c_j) = \left\|\widehat{\boldsymbol{M}}_{s_i*} - c_j\right\|_2$ is the Euclidean distance.
**(4)** Partition by assigning each state to its closest center.

## Theory

★ **Down sampling size:** By choosing block length $\tau$:

$$\tau \geq \left[\frac{2}{\Phi^2}\log\left(\sqrt{\frac{\mu_{\max}}{\mu_{\min}}}\frac{1}{\eta}\right)\right],$$

the data samples are sufficiently close to i.i.d. samples drawn from the stationary distribution of the Markov chain.

★ **Convergence analysis for GHA:**

**Principle Angle**: Given two matrices $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{m\times r}$ with orthonormal columns, the principle angle between $U$ and $V$ is defined as:

$$\Theta(U, V) = \mathrm{diag}\left[\cos^{-1}(\sigma_1(U^\top V)), \ldots, \cos^{-1}(\sigma_r(U^\top V))\right]$$

- **ODE** Characterization: **Global convergence!**

$$\text{Discrete: } \frac{\gamma_{i,t+\eta}^2 - \gamma_{i,t}^2}{\eta} = \mathcal{F}_{i,t}\gamma_{i,t}^2 + O(\eta).$$

$$\text{weakly} \Downarrow \eta \to 0$$

$$\text{Continuous: } d\gamma_i^2 = b_i\gamma_i^2 dt$$

As $\eta \to 0$ and $t \to \infty$,

$$\|\sin\Theta(\boldsymbol{V}, \widehat{\boldsymbol{V}}(t))\|_{\mathrm{F}}^2 + \|\sin\Theta(\boldsymbol{U}, \widehat{\boldsymbol{U}}(t))\|_{\mathrm{F}}^2 \to 0.$$

- **SDE** Characterization: **Convergence rate!**

$$\text{Discrete: } \frac{\zeta_{ij,t+1} - \zeta_{ij,t}}{\sqrt{\eta}} = \mathcal{F}_{ij,t}\zeta_{ij,t} + O(\eta).$$

$$\text{weakly} \Downarrow \eta \to 0$$

$$\text{Continuous: } d\zeta_{ij} = K_{ij}\zeta_{ij}dt + G_{ij}dB_t,$$

For sufficiently small $\epsilon > 0$, let

$$N = \widetilde{\mathcal{O}}\left(\frac{r}{\epsilon(\sigma_r(\boldsymbol{DP}) - \sigma_{r+1}(\boldsymbol{DP}))^2}\right) \text{ and } t = N\eta,$$

we have

$$\lim_{\epsilon \to 0} \mathbb{P}[\|\sin\Theta\left(\widehat{\boldsymbol{U}}(t), \boldsymbol{U}\right)\|_{\mathrm{F}}^2 + \|\sin\Theta\left(\widehat{\boldsymbol{V}}(t), \boldsymbol{V}\right)\|_{\mathrm{F}}^2 > \epsilon] \leq \frac{1}{10}$$

★ **Recovery of network partition:** Suppose that the estimate $\widehat{\boldsymbol{U}}$, $\widehat{\boldsymbol{V}}$, and empirical distribution $\widehat{\boldsymbol{\mu}}$ satisfy

$$\left\|\sin\Theta(\widehat{\boldsymbol{U}}, \boldsymbol{U})\right\|_F^2 + \left\|\sin\Theta(\widehat{\boldsymbol{V}}, \boldsymbol{V})\right\|_F^2 \leq \epsilon \text{ and}$$

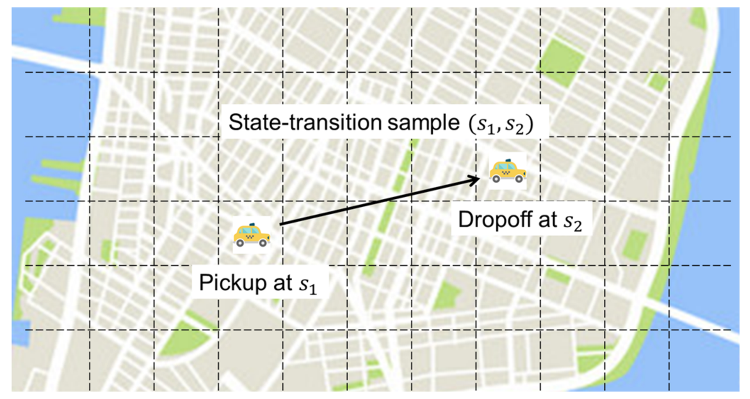
$$\max_{i \in [m]} |\widehat{\mu}_i - \mu_i| \leq \sqrt{\epsilon}\mu_i.$$

for some $\epsilon \in (0, 1)$. Let $\widehat{\boldsymbol{M}} := \mathrm{diag}(\widehat{\boldsymbol{\mu}})^{-1}\widehat{\boldsymbol{V}}$ and $\boldsymbol{M} = \boldsymbol{D}^{-1}\boldsymbol{V}$. Then for any $s_i, s_j \in S$,

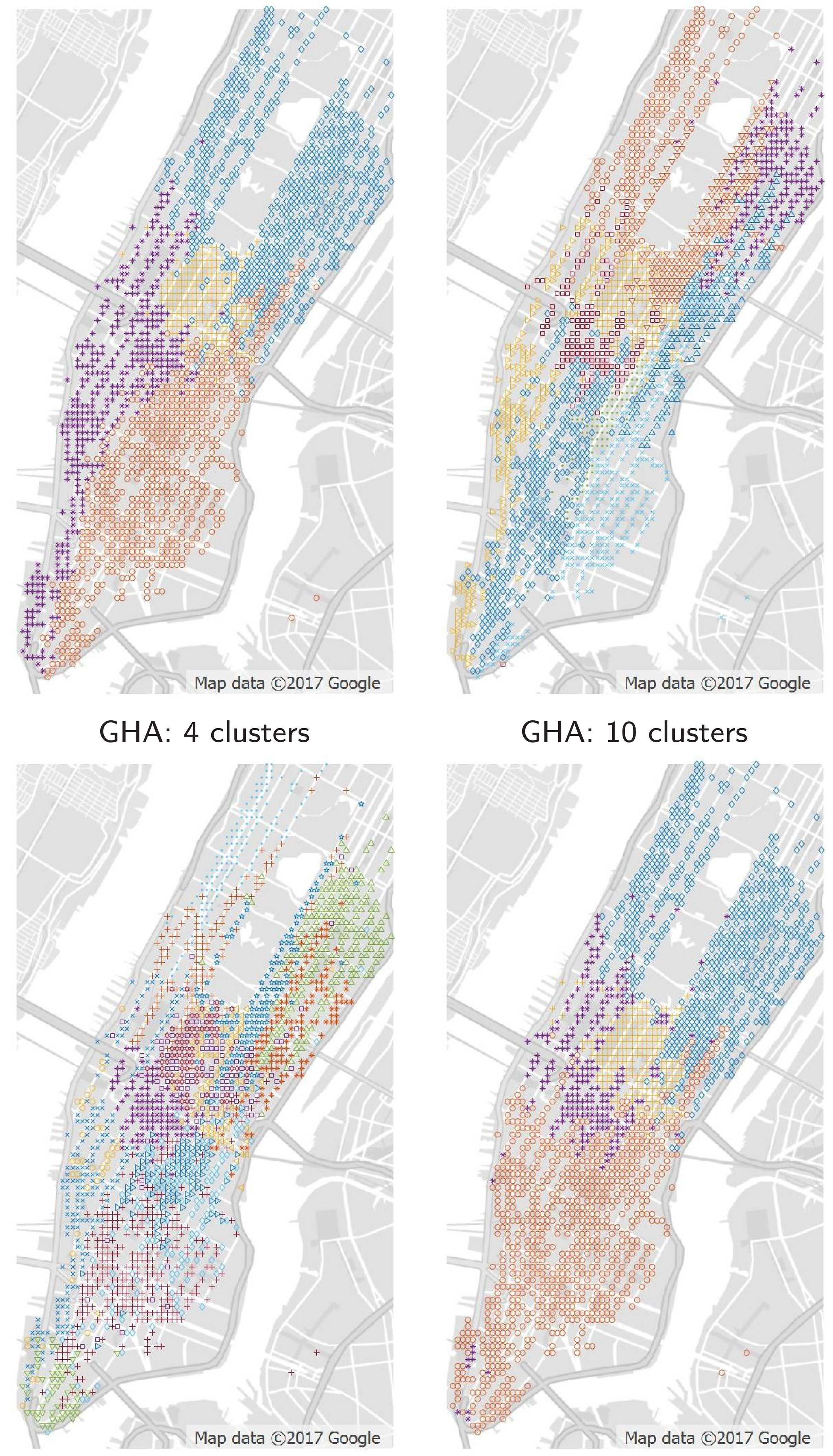
$$\left|\left\|\widehat{\boldsymbol{M}}_{s_i*} - \widehat{\boldsymbol{M}}_{s_j*}\right\|_2^2 - \left\|\boldsymbol{M}_{s_i*} - \boldsymbol{M}_{s_j*}\right\|_2^2\right| \leq \frac{C\epsilon}{\mu_{\min}^2}.$$

## Experiments

We look at Manhattan taxi data with $1.1 \times 10^7$ trip records of NYC Yellow cabs from January 2016. Each entry records the coordinates of the pick-up and drop-off locations.




We discretize the map into a fine grid and model each taxi trip as a single state transition sample generated by an implicit city-wide random walk.




GHA: 4 clusters | GHA: 10 clusters | GHA: 15 clusters | SVD: 4 clusters

A practically impressive observation is: our algorithm uses less than 1 Mbytes memory for $r = 4, 10, 15$. In contrast, the batch partition uses about 200 Mbytes memory even for $r = 4$.