北京大學
PEKING UNIVERSITY

Towards Decentralized Optimization over Digraphs: Effective metrics, lower bound, and optimal algorithms

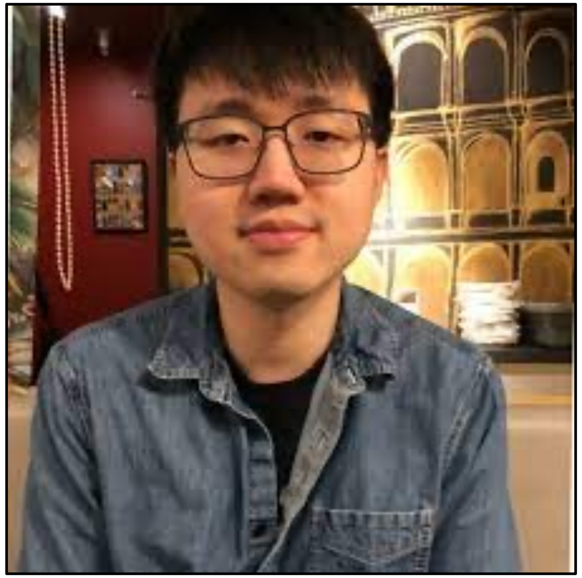
Kun Yuan

Center for Machine Learning Research @ Peking University

Joint work with

Liyuan Liang (PKU)

Xinmeng Huang (UPenn)



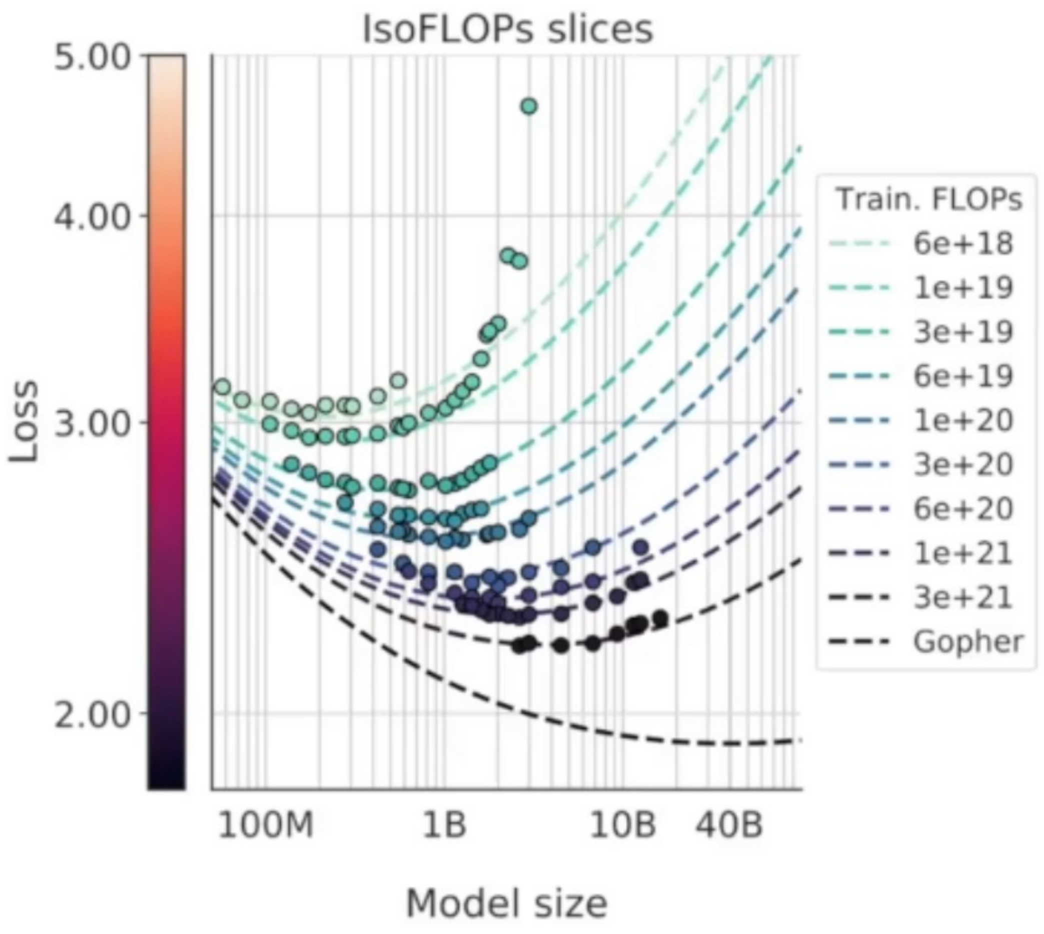
Ran Xin (ByteDance)

Part 01

Distributed training is a must in LLM

Chinchilla law

[Training Compute-Optimal Large Language Models, 2022]

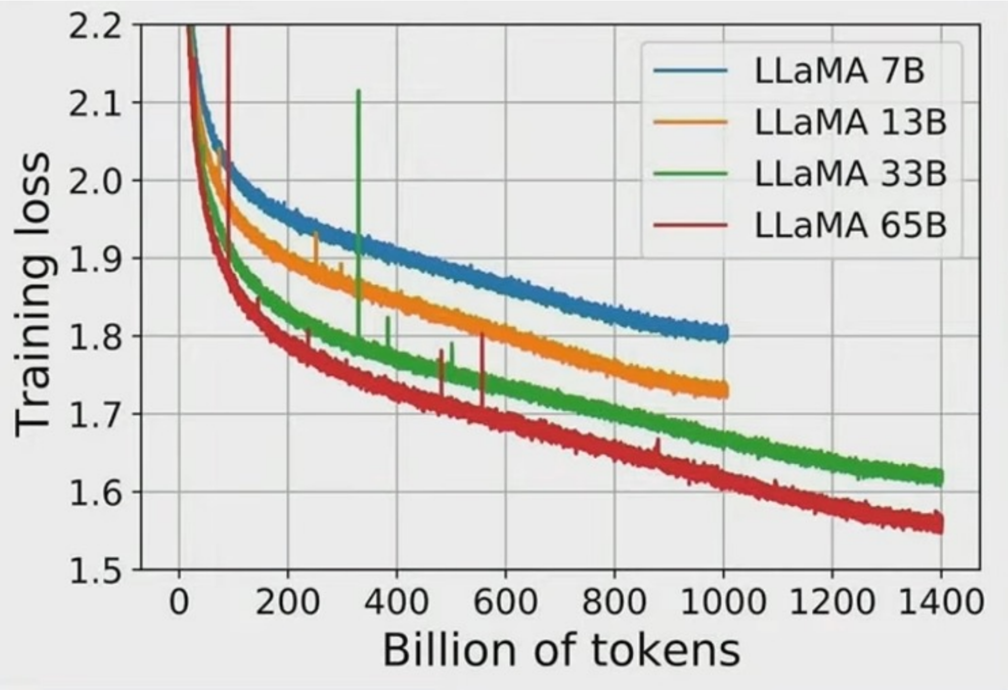


The "RGB" elements in LLM:

Larger dataset + **Bigger model** + **Longer training** = **Better LLM**

LLaMA follows Chinchilla law

[LLaMA: Open and Efficient Foundation Language Models,2023]



According to Chinchilla law:

- LLM model gets increasingly bigger
- Dataset gets increasingly larger
- Distributed training gets increasingly important

Thousands of GPUs are needed to train LLM

[State of GPT, 2023]

2 example models

GPT-3 (2020)

50,257 vocabulary size
2048 context length
175B parameters
Trained on 300B tokens

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Training: (rough order of magnitude to have in mind)

- O(1,000 - 10,000) V100 GPUs
- O(1) month of training
- O(1-10) $M

LLaMA (2023)

32,000 vocabulary size
2048 context length
65B parameters
Trained on 1-1.4T tokens

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

Training for 65B model:

- 2,048 A100 GPUs
- 21 days of training
- $5M

Recent GPU clusters scale with more than 10000 GPUs

北京大学 PEKING UNIVERSITY

Feb. 23, 2024

MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs

Ziheng Jiang[1,*] Haibin Lin[1,*] Yinmin Zhong[2,*] Qi Huang[1] Yangrui Chen[1] Zhi Zhang[1]
Yanghua Peng[1] Xiang Li[1] Cong Xie[1] Shibiao Nong[1] Yulu Jia[1] Sun He[1] Hongmin Chen[1]
Zhihao Bai[1] Qi Hou[1] Shipeng Yan[1] Ding Zhou[1] Yiyao Sheng[1] Zhuo Jiang[1]
Haohan Xu[1] Haoran Wei[1] Zhang Zhang[1] Pengfei Nie[1] Leqi Zou[1] Sida Zhao[1]
Liang Xiang[1] Zherui Liu[1] Zhe Li[1] Xiaoying Jia[1] Jianxi Ye[1] Xin Jin[2,†] Xin Liu[1,†]

[1]*ByteDance* [2]*Peking University*

Abstract

We present the design, implementation and engineering experience in building and deploying MegaScale, a production system for training large language models (LLMs) at the scale of more than 10,000 GPUs. Training LLMs at this scale brings unprecedented challenges to training efficiency and stability. We take a full-stack approach that co-designs the algorithmic and system components across model block and optimizer design, computation and communication overlapping, oper-

serving billions of users, we have been aggressively integrating AI into our products, and we are putting LLMs as a high priority to shape the future of our products.

Training LLMs is a daunting task that requires enormous computation resources. The scaling law [3] dictates that the model size and the training data size are critical factors that determine the model capability. To achieve state-of-the-art model capability, many efforts have been devoted to train large models with hundreds of billions or even trillions of parameters on hundreds of billions or even trillions of tokens. For example, GPT-3 [4] has 175 billion parameters and

Llama 3 uses 24000 GPUs in LLM training

Apr. 18, 2024

Build the future of AI with Meta Llama 3

To train our largest Llama 3 models, we combined three types of parallelization: data parallelization, model parallelization, and pipeline parallelization. Our most efficient implementation achieves a compute utilization of over 400 TFLOPS per GPU when trained on 16K GPUs simultaneously. We performed training runs on two custom-built 24K GPU clusters. To maximize GPU uptime, we developed an advanced new training stack that automates error detection, handling, and maintenance. We also greatly improved our

[Introducing Meta Llama 3: The most capable openly available LLM to date]

Distributed training over massive GPUs is extremely challenging

- Distributed training over thousands of GPUs is extremely challenging
- Two major challenges: **stability** and **scalability**
- Stability: very common that some GPU crashes during training LLMs
  - Meta OPT-175B: 175+ job restarts caused by hardware failures in 2 months
  - 书生大模型: Waste 41700 GPU hours due to training crashes (>80% are infrastructure failures)
- Stability in LLM training is very important, but this talk will not discuss it

Distributed training over massive GPUs is extremely challenging

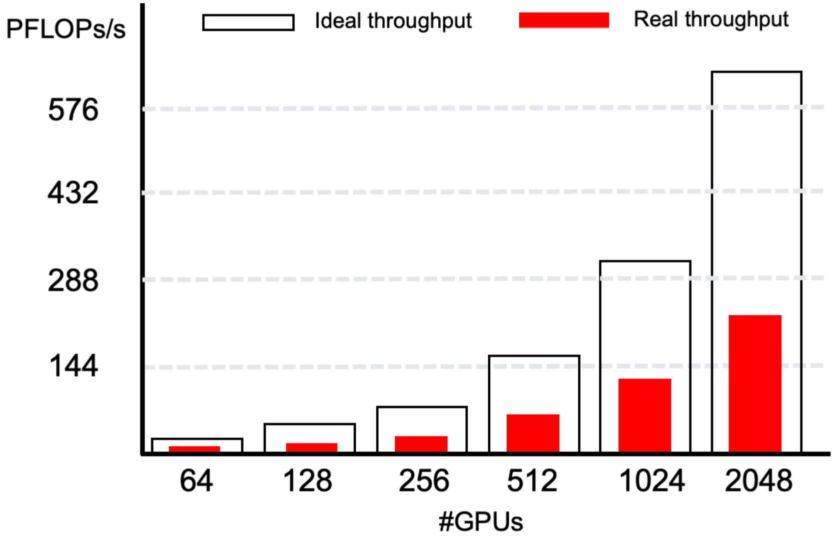
- The **communication overhead** and **GPU idle time** severely hamper the scalability

- Each GPU can only achieve **30%~55%** of its peak FLOPs/s during LLM training

- When GPU achieves 30% of peak FLOP/s, we say the system achieves 30% scalability

  - Nvidia Megatron achieves **~50%** scalability
  - OpenAI GPT-3 achieves **~44%** scalability
  - Meta LLaMA-65B achieves **~30%** scalability

30% of its peak FLOPs/s visualization



Existing systems suffer from severe scalability issue

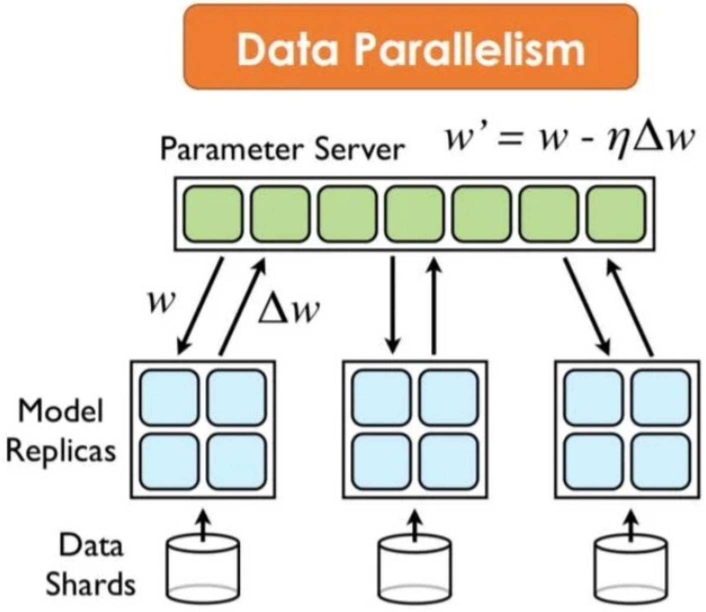
[Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM, 2021]

Number of parameters (billion)	Attention heads	Hidden size	Number of layers	Tensor model-parallel size	Pipeline model-parallel size	Number of GPUs	Batch size	Achieved teraFlOP/s per GPU	Percentage of theoretical peak FLOP/s	Achieved aggregate petaFLOP/s
1.7	24	2304	24	1	1	32	512	137	44%	4.4
3.6	32	3072	30	2	1	64	512	138	44%	8.8
7.5	32	4096	36	4	1	128	512	142	46%	18.2
18.4	48	6144	40	8	1	256	1024	135	43%	34.6
39.1	64	8192	48	8	2	512	1536	138	44%	70.8
76.1	80	10240	60	8	4	1024	1792	140	45%	143.8
145.6	96	12288	80	8	8	1536	2304	148	47%	227.1
310.1	128	16384	96	8	16	1920	2160	155	50%	297.4
529.6	128	20480	105	8	35	2520	2520	163	52%	410.2
1008.0	160	25600	128	8	64	3072	3072	163	52%	502.0

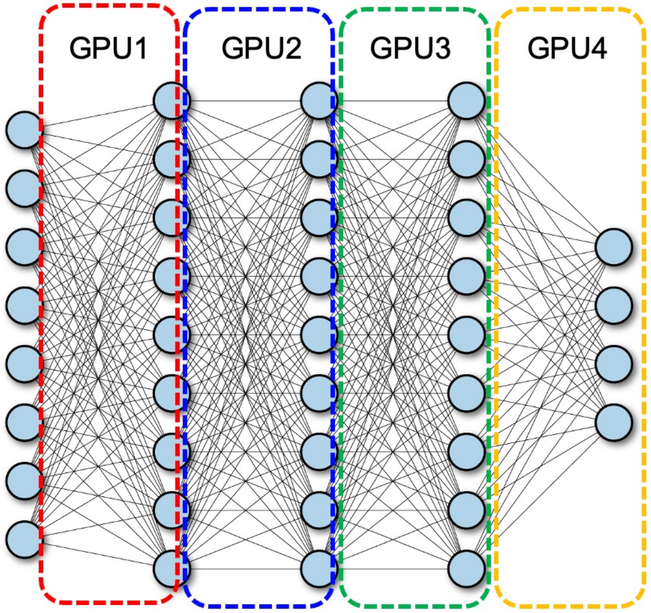
Nvidia Megatron: the most popular distributed training framework

Communication overhead is the top factor hampering the scalability

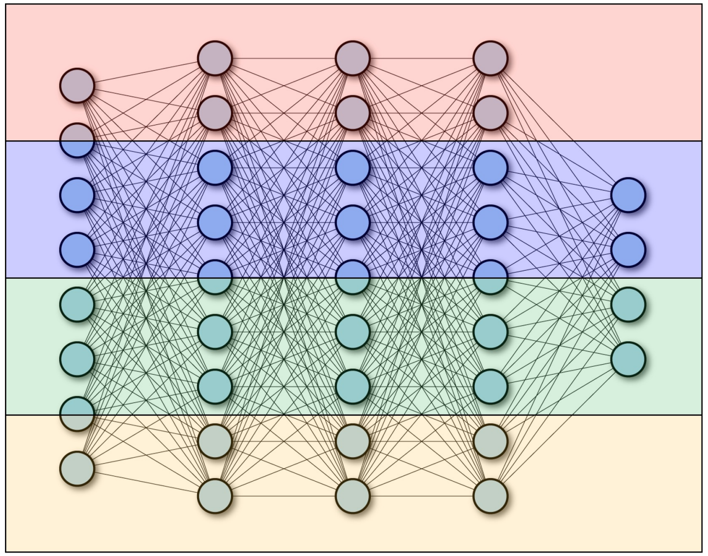
- 3D parallelism indicates 3 orthogonal parallel techniques used to train LLM



Data parallelism



Pipeline parallelism



Tensor parallelism

- This talk mainly focuses on **data parallelism**.

Part 02

Decentralized optimization

Data-parallel distributed learning

北京大學 PEKING UNIVERSITY

- Training deep neural networks typically requires **massive** datasets; efficient and scalable distributed optimization algorithms are in urgent need

- A network of n nodes (devices such as GPUs) collaborate to solve the problem:

$$\min_{x \in \mathbb{R}^d} \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \quad \text{where} \quad \boxed{f_i(x)} = \boxed{\mathbb{E}_{\xi_i \sim D_i} F(x;\xi_i)}.$$

  - Each component $f_i : \mathbb{R}^d \to \mathbb{R}$ is local and private to node i
  - Random variable ξ_i denotes the local data that follows distribution D_i
  - Each local distribution D_i is different; data heterogeneity exists

Vanilla parallel stochastic gradient descent (PSGD)

$$g_i^{(k)} = \nabla F(x^{(k)}; \xi_i^{(k)}) \qquad \text{(Local compt.)}$$

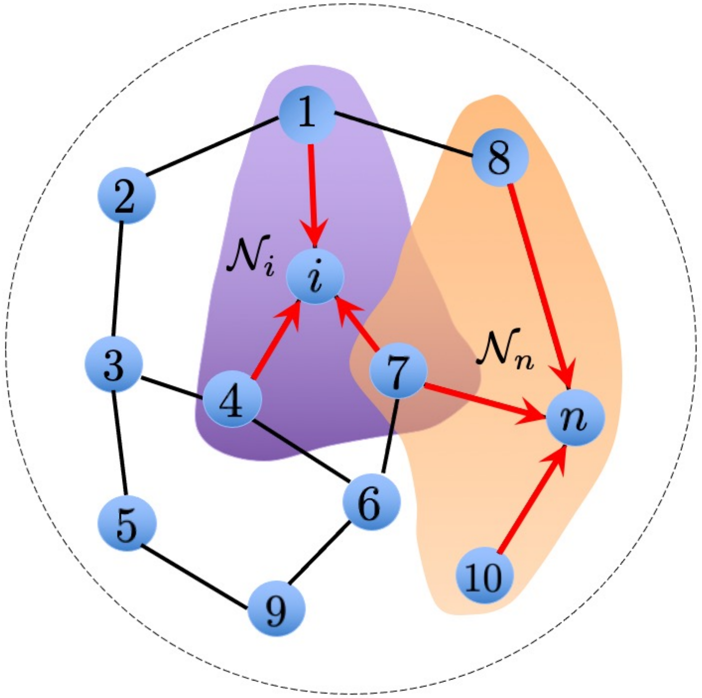
$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n} \sum_{i=1}^{n} g_i^{(k)} \qquad \text{(Global comm.)}$$

- Each node i samples data $\xi_i^{(k)}$ and computes gradient $\nabla F(x^{(k)}; \xi_i^{(k)})$
- All nodes synchronize (i.e. globally average) to update model x per iteration
- PSGD is communication-expensive due to the existence of the global averaging

Decentralized SGD (DSGD)

- DSGD = local SGD update + partial averaging [LS08]

$$x_i^{(k+\frac{1}{2})} = x_i^{(k)} - \gamma \nabla F(x_i^{(k)}; \xi_i^{(k)}) \quad \text{(Local update)}$$
$$x_i^{(k+1)} = \sum_{\boxed{j \in \mathcal{N}_i}} \boxed{w_{ij}} x_j^{(k+\frac{1}{2})} \quad \text{(Partial averaging)}$$



- $\mathcal{N}_i$ is the set of neighbors at node i ;
- w_{ij} scales information from j to i and satisfies $\sum_{j \in \mathcal{N}_i} w_{ij} = 1$
- Communication-efficient by removing the global average

DSGD is more communication-efficient than PSGD

- A real experiment on a 256-GPUs cluster [CYZ+21]

Model	Ring-Allreduce	Partial average
ResNet-50 (25.5M)	278 ms	150 ms
Bert (300M)	1469 ms	567 ms

Table. Comparison of per-iter comm. time in terms of runtime with 256 GPUs

- DSGD saves more communications per iteration for larger models

[CYZ+21] Y. Chen*, K. Yuan*, Y. Zhang, P. Pan, Y. Xu, and W. Yin, ``Accelerating Gossip SGD with Periodic Global Averaging”, ICML 2021

Does decentralized SGD converge? Yes!

PEKING UNIVERSITY 北京大学

- Recall the PSGD and DSGD recursions

Parallel SGD

$$x_i^{(k+\frac{1}{2})} = x_i^{(k)} - \gamma \nabla F(x_i^{(k)}; \xi_i^{(k)}) \quad \text{(Local update)}$$

$$x_i^{(k+1)} = \frac{1}{n}\sum_{j=1}^{n} x_j^{(k+\frac{1}{2})} \quad \text{(Global averaging)}$$

Decentralized SGD

$$x_i^{(k+\frac{1}{2})} = x_i^{(k)} - \gamma \nabla F(x_i^{(k)}; \xi_i^{(k)}) \quad \text{(Local update)}$$

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(k+\frac{1}{2})} \quad \text{(Partial averaging)}$$

- DSGD will converge if the **partial average** asymptotically converge to the **global average**

- This argument is true if the weight matrix $W = [w_{ij}]_{i=1,j=1}^{n} \in \mathbb{R}^{n\times n}$ is doubly-stochastic

$$W\mathbf{1}_n = \mathbf{1}_n \quad \text{and} \quad \mathbf{1}_n^T W = \mathbf{1}_n^T$$

Does decentralized SGD converge? Yes!

- If the matrix is **doubly-stochastic**, it holds that

$$W\mathbf{1}_n = \mathbf{1}_n \quad \text{and} \quad \mathbf{1}_n^T W = \mathbf{1}_n^T \quad \Longrightarrow \quad W^k \longrightarrow \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T \qquad (k \to \infty)$$

- Given a doubly-stochastic W, the **partial average** will converge to the global average

  - We introduce $\mathbf{z} = [z_1^\top; z_2^\top; \cdots; z_n^\top] \in \mathbb{R}^{n\times d}$ and $\mathbf{z}^{(0)} = \mathbf{z}$, it holds that

$$\mathbf{z}^{(k)} = W\mathbf{z}^{(k-1)} = W^k\mathbf{z}^{(0)} \longrightarrow \quad (1/n)\mathbf{1}_n\mathbf{1}_n^T\mathbf{z}^{(0)} \qquad (\text{as } k \to \infty)$$

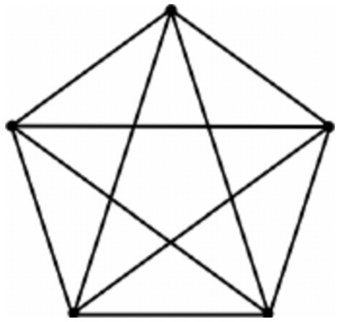
  - This implies that $z_i^{(k)} \to (1/n)\sum_{j=1}^n z_j$

$$\boxed{\begin{aligned} x_i^{(k+\frac{1}{2})} &= x_i^{(k)} - \gamma \nabla F(x_i^{(k)}; \xi_i^{(k)}) && \text{(Local update)} \\ x_i^{(k+1)} &= \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(k+\frac{1}{2})} && \text{(Partial averaging)} \end{aligned}}$$

Doubly-stochastic matrix is easy to construct over undirected graph

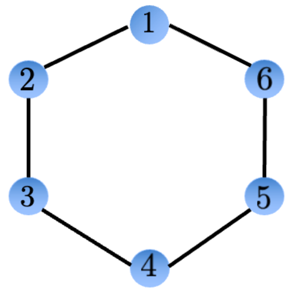
- Doubly-stochastic weight matrix is easy to construct over **undirected** graphs

Fully-connected graph



$$W = \frac{1}{5}\mathbb{1}\mathbb{1}^T = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}$$

Ring graph

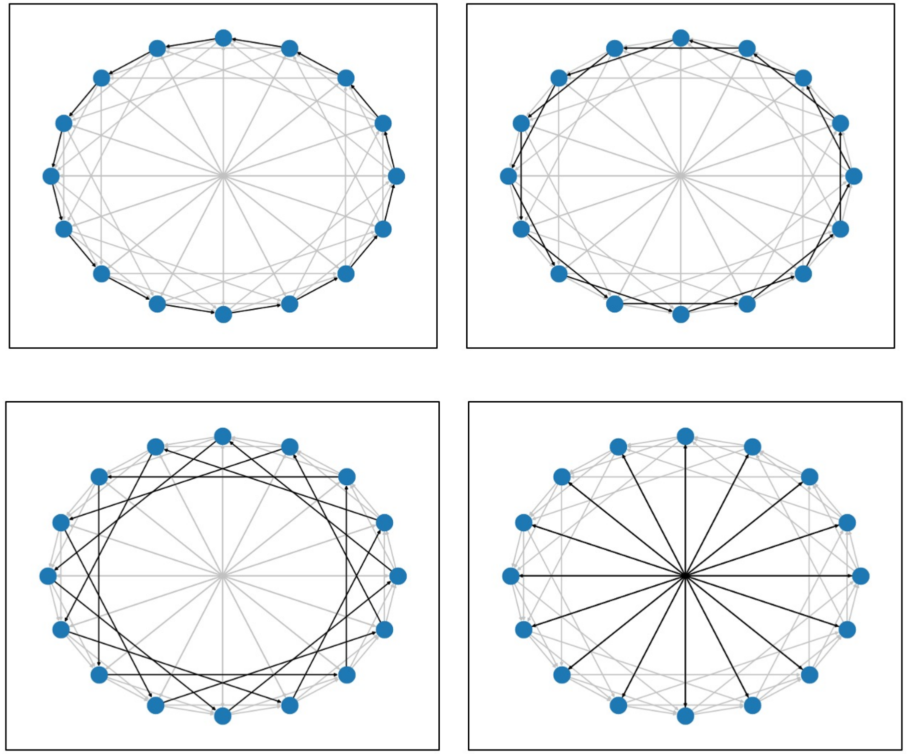


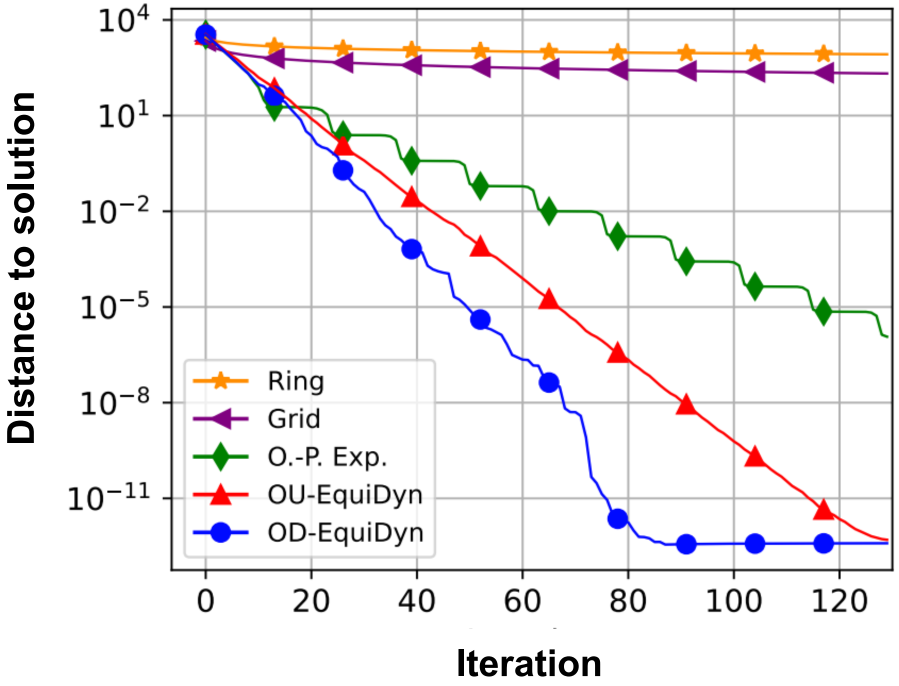
$$W = \begin{bmatrix} 1/3 & 1/3 & 0 & 0 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 0 & 0 & 1/3 & 1/3 \end{bmatrix}$$

$$w_{ij} = \begin{cases} 1/(1+d_i) & \text{if node } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

Network topology determines the convergence rate

- Network topology determines how fast that DSGD will converge



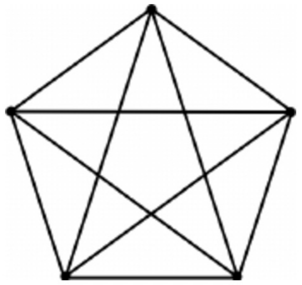


Influence of undirected network topology

- Influence of undirected network is well studied in literature

- We introduce quantity ρ to gauge the graph connectivity

$$\rho = \|W - \frac{1}{n}\mathbb{1}\mathbb{1}^T\|_2 \in (0,1) \ \text{ where } W = [w_{ij}] \in \mathbb{R}^{n\times n}$$



$$W = \frac{1}{5}\mathbb{1}\mathbb{1}^T = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}$$

  - Well-connected topology has $\rho \to 0$, e.g. fully-connected topology
  - Sparsely-connected topology has $\rho \to 1$, e.g. ring has $\rho = O(1 - \frac{1}{n^2})$

Influence of undirected network topology

- Convergence comparison (non-convex and **data-homogeneous** scenario) [KLB+20]:

$$\text{P-SGD}: \quad \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\bar{x}^{(k)})\|^2 = O\Big(\boxed{\frac{\sigma}{\sqrt{nT}}}\Big)$$

$$\text{D-SGD}: \quad \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\bar{x}^{(k)})\|^2 = O\Big(\boxed{\frac{\sigma}{\sqrt{nT}}} + \underbrace{\boxed{\frac{\rho^{2/3}\sigma^{2/3}}{T^{2/3}(1-\rho)^{1/3}}}}_{\text{extra overhead}}\Big)$$

where σ^2 is the gradient noise, and T is the number of iterations

- D-SGD can asymptotically converge as fast as P-SGD when $T \to \infty$; the first term dominates; reach **linear speedup** asymptotically

- But sparse topology ($\rho \to 1$) will significantly slow down the convergence of DSGD

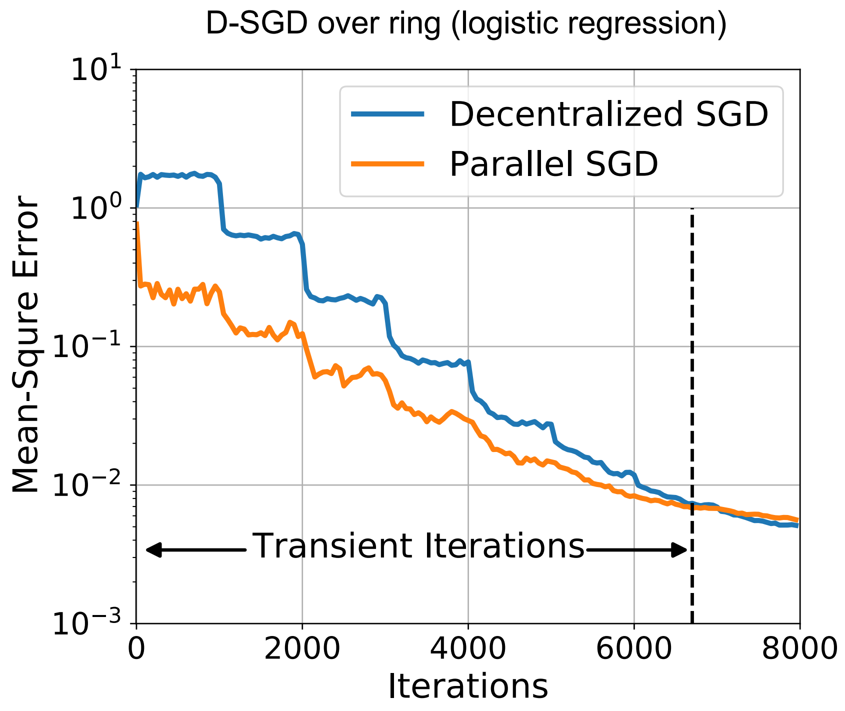
Transient iterations

- Definition [POP21]: number of iterations before D-SGD achieves linear speedup

- D-SGD for non-convex and data-homogeneous scenario has $O(n^3(1-\rho)^{-2})$ transient iterations

$$\frac{\rho^{2/3}\sigma^{2/3}}{T^{2/3}(1-\rho)^{1/3}} \le \frac{\sigma}{\sqrt{nT}} \quad \Longrightarrow \quad O(\frac{\rho^4 n^3}{(1-\rho)^2})$$

- Sparse topology $\rho \to 1$ incurs longer tran. Iters.



Techniques to reduce transient iterations

- Remove the influence of data heterogeneity
  - Exact-Diffusion [YAYS20, YAH23] (also known as D2 [TLY+18])
  - Gradient tracking [KLS22, AY22]
- Develop multi-step gossip strategy to accelerate convergence
  - DeTAG [LS21]; MG-DSGD[YHC+22]; MG-Exact-Diffusion[YAH23]
- Develop sparse and effective network topologies
  - Exponential Graph [YYC+21]; EquiTopo [SLJ+22]
  - CECA-DSGD [DJY+23]

Algorithm	Tran. Iters.
DSGD	$\mathcal{O}(n^3/(1-\rho)^4)$
+ ED/D2	$\mathcal{O}(n^3/(1-\rho)^2)$
+ MG	$\mathcal{O}(n/(1-\rho))$
+ EquiTopo	$\mathcal{O}(n)$

The key step is to **identify an effective metric** that captures the influence of network topology

北京大学
PEKING UNIVERSITY

Part 03

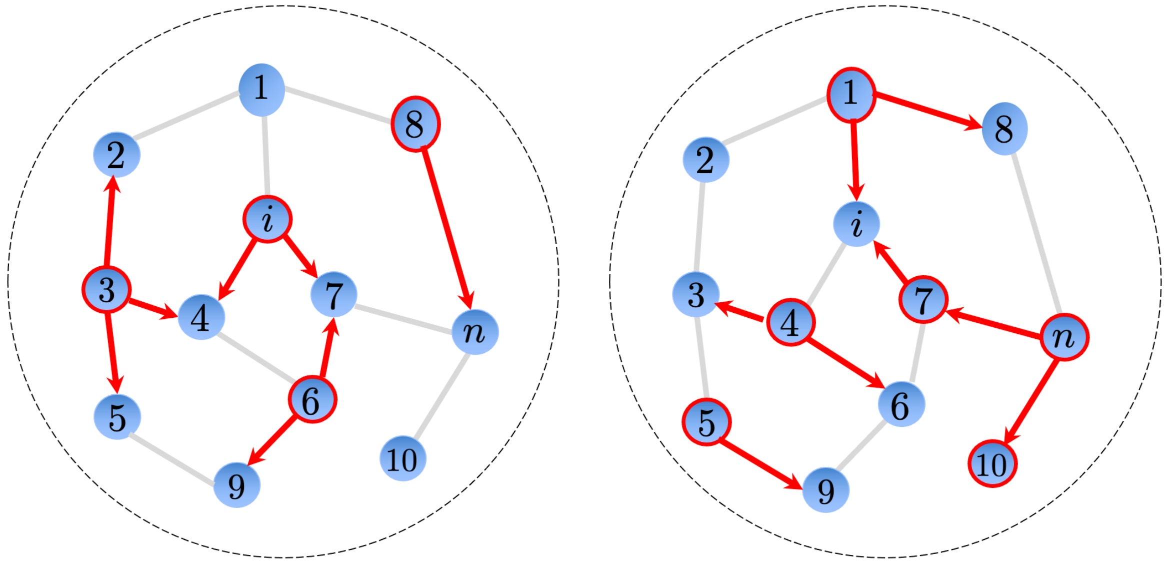
Decentralized Optimization over Digraphs

Directed network topology

Social networks are directed

Asynchronous networks are directed

(nodes are activated at different iteration)



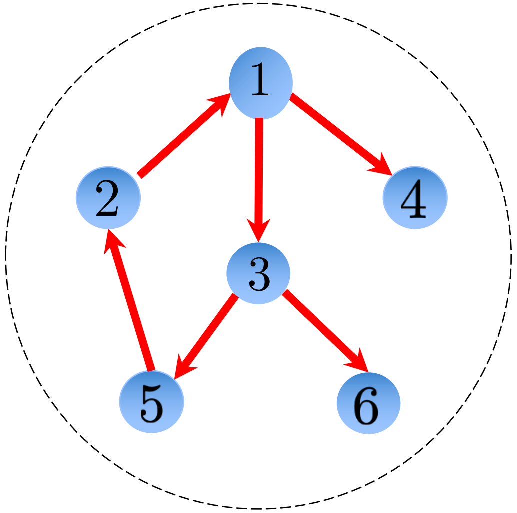
iteration k

iteration k+1

Column-stochastic weight matrix

- This talk focuses on the **column-stochastic** weight matrix associated with a directed network
- A common way to construct the column-stochastic weight matrix

$$w_{ij} = \begin{cases} 1/(1+d_i^{\text{out}}) & \text{if directed edge } (j,i) \in \mathcal{E} \text{ or } j = i \\ 0 & \text{otherwise} \end{cases}$$



$$\begin{bmatrix} \frac{1}{3} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{3} & 0 & 0 & 1 \end{bmatrix}$$

Column-stochastic weight matrix

Assumption 2.

The mixing matrix W is entry-wisely non-negative, primitive (i.e., all entries of W^{k_0} are positive for sufficiently large $k_0 \in \mathbb{N}_+$), and satisfies $\mathbf{1}_n^\top W = \mathbf{1}_n^\top$.

Lemma 2.

Under Assumption 2, there exists a unique equilibrium vector $\pi \in \mathbb{R}^n$ with positive entries such that $W\pi = \pi$ and $\mathbf{1}_n^\top \pi = 1$. Moreover, it holds that

$$\lim_{k\to\infty} W^k = \pi \mathbf{1}_n^\top$$

For doubly stochastic W, it holds that $\pi = (\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})$, which reduces to $W \to \frac{1}{n}\mathbf{1}\mathbf{1}^T$

Column-stochastic matrix cannot enable global average

- Assume each node i maintains a local vector $z_i \in \mathbb{R}^n$

- We introduce $\mathbf{z} = [z_1^\top; z_2^\top; \cdots; z_n^\top] \in \mathbb{R}^{n\times d}$ and let $\mathbf{z}^{(0)} = \mathbf{z}$

- Column-stochastic matrix cannot enable global average

$$\mathbf{z}^{(k)} = W\mathbf{z}^{(k-1)} = W^k\mathbf{z}^{(0)} \longrightarrow \pi\mathbf{1}^\top\mathbf{z}^{(0)} \quad (\text{as } k \to \infty)$$

which implies that $z_i^{(k)} \to \pi_i \sum_{j=1}^n z_j$ rather than $z_i^{(k)} \to (1/n)\sum_{j=1}^n z_j$

- We can correct the bias to enable global average

$$\mathbf{w}^{(k)} = \mathrm{diag}(n\pi)^{-1}\mathbf{z}^{(k)} \longrightarrow \mathrm{diag}(n\pi)^{-1}\pi\mathbf{1}_n^\top\mathbf{z} = (1/n)\mathbf{1}_n\mathbf{1}_n^\top\mathbf{z}$$

Push-sum decentralized averaging

(Bias correction) $\mathbf{w}^{(k)} = \mathrm{diag}(n\pi)^{-1}\mathbf{z}^{(k)} \longrightarrow \mathrm{diag}(n\pi)^{-1}\pi\mathbf{1}_n^\top\mathbf{z} = (1/n)\mathbf{1}_n\mathbf{1}_n^\top\mathbf{z}$

- However, the equilibrium vector π is not known in advance
- **Push-sum** decentralized averaging [BBT+10, TLR12, NO13]

$$\begin{aligned}\mathbf{z}^{(k+1)} &= W\mathbf{z}^{(k)} \\ v^{(k+1)} &= Wv^{(k)} \\ V^{(k+1)} &= \mathrm{diag}(v^{(k+1)}) \\ \mathbf{w}^{(k+1)} &= {V^{(k+1)}}^{-1}\mathbf{z}^{(k+1)}.\end{aligned}$$

(starting with $v^{(0)}$ satisfying $\mathbf{1}_n^\top v^{(0)} = n$)

- It is guaranteed that $w_i^{(k)} \to (1/n)\sum_{j=1}^n z_j$

Push-sum decentralized optimization

PEKING UNIVERSITY

- Many optimization algorithms over digraphs have been proposed based on push-sum averaging
  - Push-sum subgradient method [NO13]
  - Push-sum dual averaging [TLR12]
  - Push-sum EXTRA [ZY17; XK17]
  - Push-sum Gradient-tracking [NOS17]
  - Push-sum SGD [ALBR19]

Push-sum decentralized optimization

- Existing results show that algorithms over digraphs asymptotically converge as fast as centralized algorithms

Algorithm	Rate (A.)	Rate (F.T.)	Transient Stage
Gradient-Push [4]	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	N.A.	N.A.
Push-DIGing [15]	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	N.A.	N.A.

- Open question:
  - How much slower decentralized optimization over digraphs compared to centralized optimization?
  - How does the digraphs influence the convergence rate?
  - How to reduce the influence of digraphs

北京大學
PEKING UNIVERSITY

Part 04

Effective metrics to evaluate digraphs

Spectral gap

- In undirected graphs, the connectivity is gauged by spectral gap $1-\beta$ where

$$\beta = \|W - (1/n)\mathbf{1}_n\mathbf{1}_n^\top\|_2 \in [0,1)$$

(Recall that $W^k \longrightarrow \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$)

- Inspired by this, can we use the a similar metric to capture the connectivity of digraphs?

$$\beta_\pi = \|W - \pi\mathbf{1}_n^\top\|_\pi \in [0,1)$$

(Recall that $W^k \to \pi\mathbf{1}_n^T$)

where for a matrix A, its π-norm is defined as

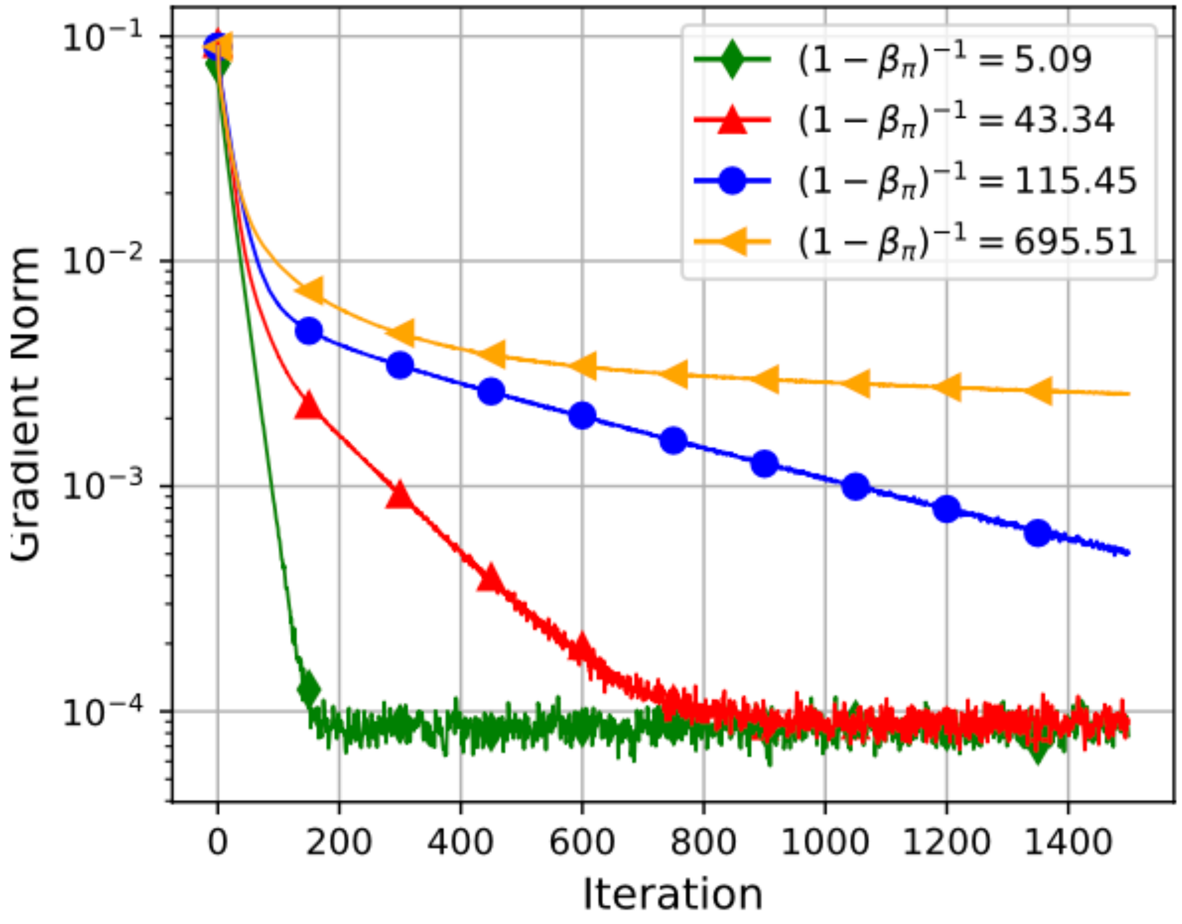
$$\|A\|_\pi := \|\mathrm{diag}(\sqrt{\pi})^{-1}A\ \mathrm{diag}(\sqrt{\pi})\|_2$$

- When the topology is undirected, it holds that $\beta_\pi = \beta$

Spectral gap alone cannot precisely reflect the digraph influence

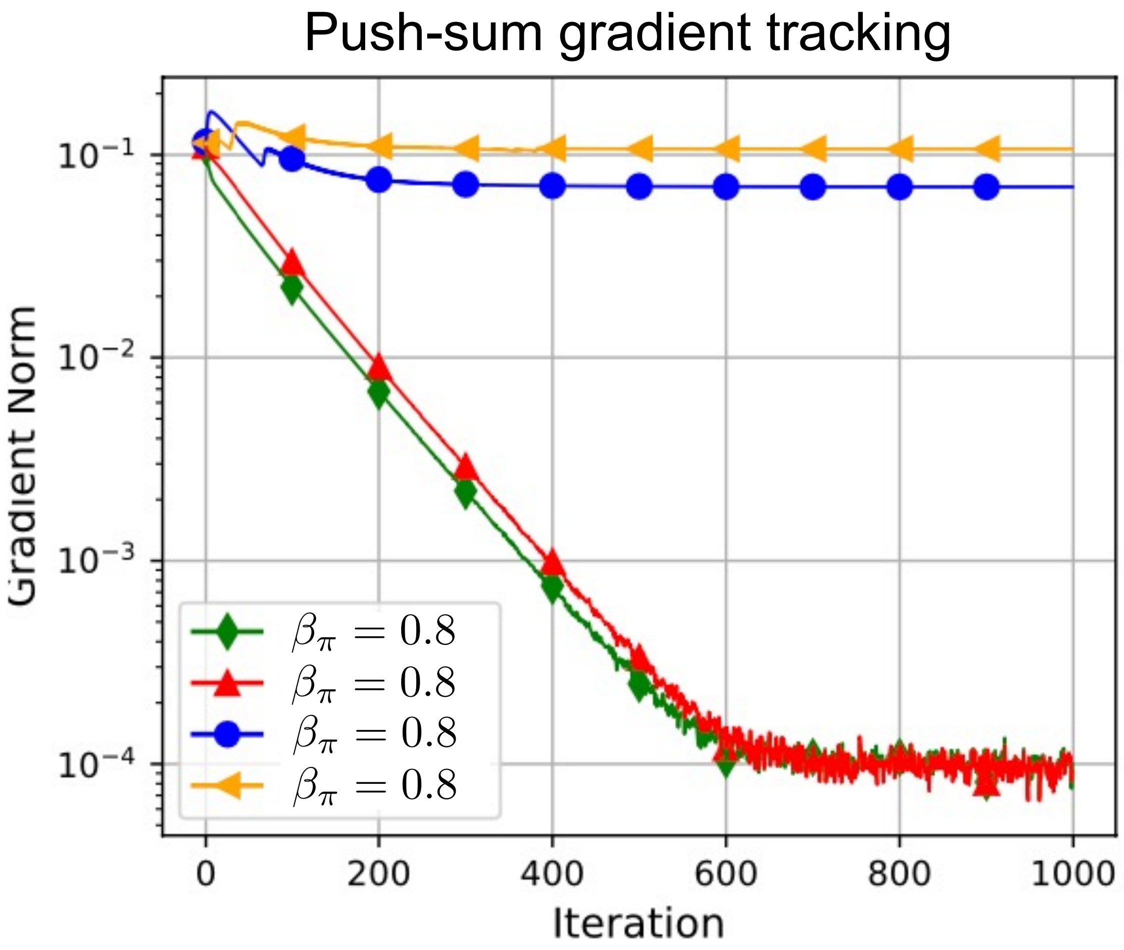
- **We are done!** We can use β_π to capture the influence of digraphs.

Push-sum gradient tracking



Spectral gap alone cannot precisely reflect the digraph influence

- Wait! Something strange is happening!
- Different curves vary drastically even with the same spectral gap!
- The single spectral gap alone is insufficient to capture the digraph influence
- **Something strange is missing!**



Equilibrium skewness

PEKING UNIVERSITY

- Let's revisit the power iteration with column-stochastic W. Initialize $x^{(0)} = x$, we have

$$x^{(k)} = Wx^{(k-1)} = W^k x^{(0)} = W^k x \longrightarrow \pi \mathbf{1}_n^\top x \quad \text{as} \quad k \to \infty.$$

- To evaluate how fast $x^{(k)}$ converges to the global average $(1/n)\mathbf{1}_n\mathbf{1}_n^\top x$:
  - Generalized spectral gap $\beta_\pi = \|W - \pi\mathbf{1}_n^\top\|_\pi$ gauges how fast that $x^{(k)}$ approaches to $\pi\mathbf{1}_n^\top x$
  - A new metric to gauge the disagreement between $\pi\mathbf{1}_n^\top x$ and $(1/n)\mathbf{1}_n\mathbf{1}_n^\top x$

Equilibrium skewness

- Let's revisit the power iteration with column-stochastic W. Initialize $x^{(0)} = x$, we have

$$x^{(k)} = Wx^{(k-1)} = W^k x^{(0)} = W^k x \longrightarrow \pi \mathbf{1}_n^\top x \quad \text{as} \quad k \to \infty.$$

- To evaluate how fast $x^{(k)}$ converges to the global average $(1/n)\mathbf{1}_n\mathbf{1}_n^\top x$:
  - Generalized spectral gap to gauge how fast that $x^{(k)}$ approaches to $\pi \mathbf{1}_n^\top x$
  - **Equilibrium skewness** to gauge the disagreement between $\pi \mathbf{1}_n^\top x$ and $(1/n)\mathbf{1}_n\mathbf{1}_n^\top x$

$$\kappa_\pi := \max_i \pi_i / \min_i \pi_i \in [1, +\infty)$$

Revisit push-sum decentralized averaging

Push-sum averaging

$$\mathbf{z}^{(k+1)} = W\mathbf{z}^{(k)}$$
$$v^{(k+1)} = Wv^{(k)}$$
$$V^{(k+1)} = \mathrm{diag}(v^{(k+1)})$$
$$\mathbf{w}^{(k+1)} = {V^{(k+1)}}^{-1}\mathbf{z}^{(k+1)}.$$

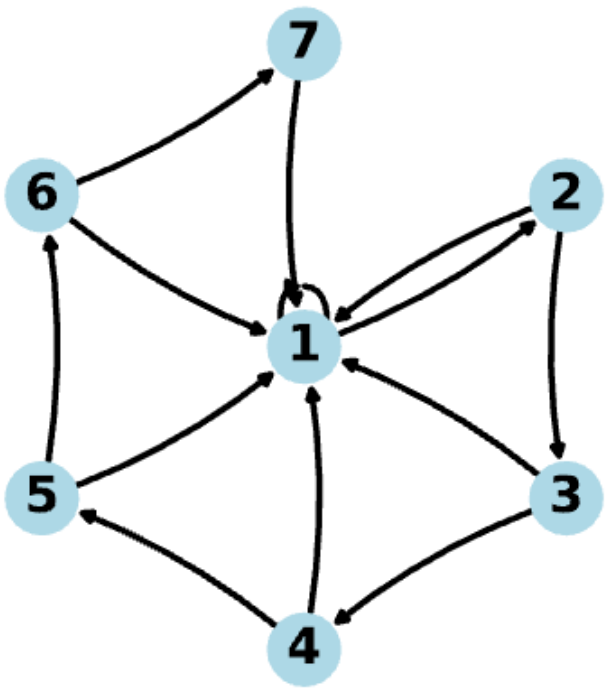
Theorem 1.

Assume $v^{(0)}$ is initialized such that $\mathbf{1}_n^\top v^{(0)} = n$, Push -sum decentralized averaging converges as follows

$$\|\mathbf{w}^{(k)} - \bar{\mathbf{z}}\|_F \le \kappa_\pi^{3/2}\beta_\pi^k\|\mathbf{z}^{(0)}\|_F$$

The **first** rate reflecting the influence of both κ_π and β_π

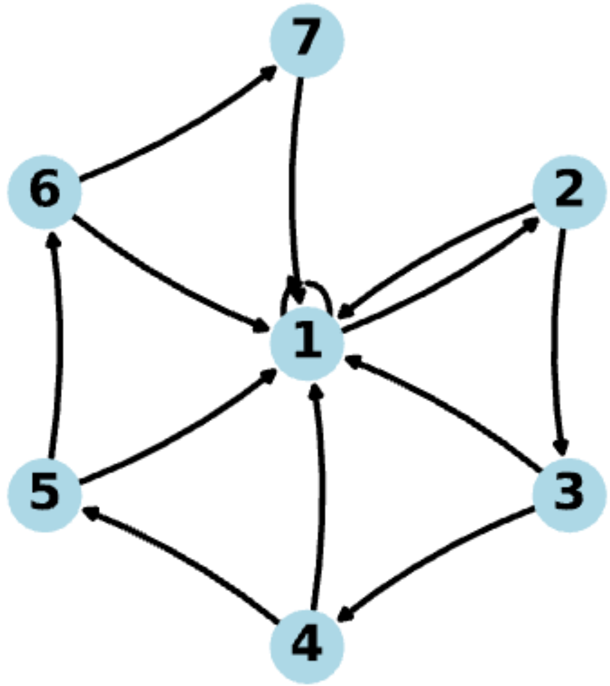
The influence of equilibrium skewness is substantial



$$W = \begin{bmatrix} 1/2 & 1/2 & \cdots & 1/2 & 1 \\ 1/2 & 0 & & & \\ & \ddots & \ddots & & \\ & & 1/2 & 0 & \\ & & & 1/2 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

The probability that a message flows from node 1 to node n decays **exponentially** fast

The influence of equilibrium skewness is substantial

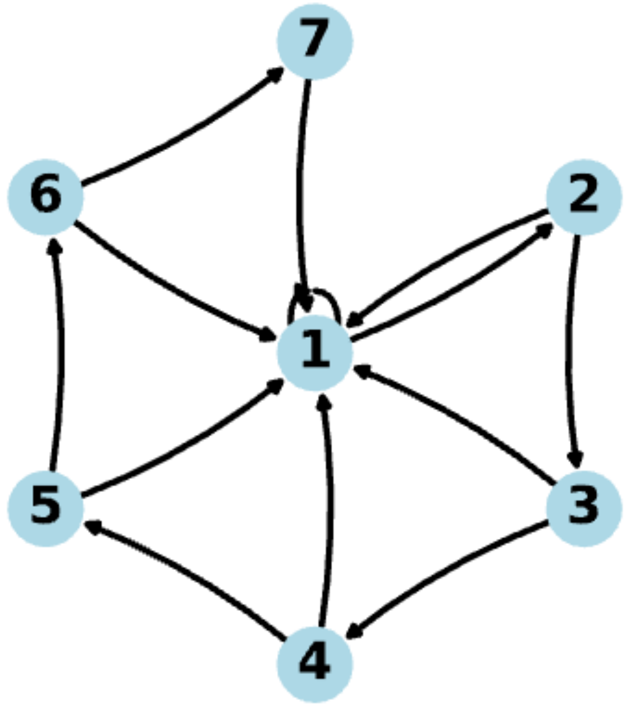


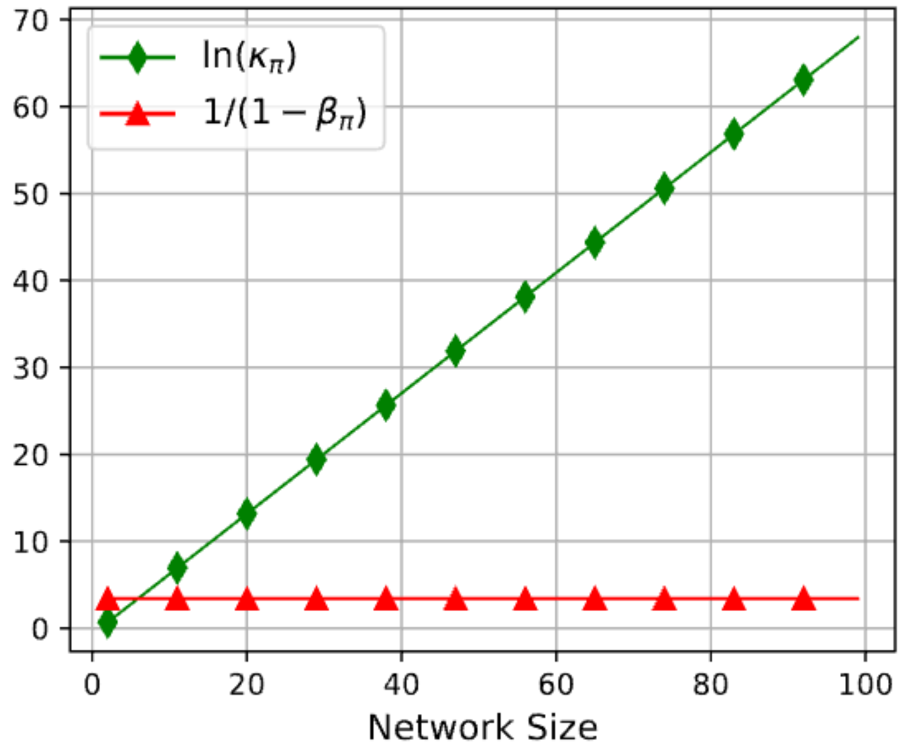
Theorem 2.

For any $n \geq 1$, there exists a matrix $W \in \mathbb{R}^{n \times n}$ such that

$$\beta_\pi = \frac{\sqrt{2}}{2} \quad \text{and} \quad \kappa_\pi = 2^{n-1}$$

The influence of equilibrium skewness is substantial



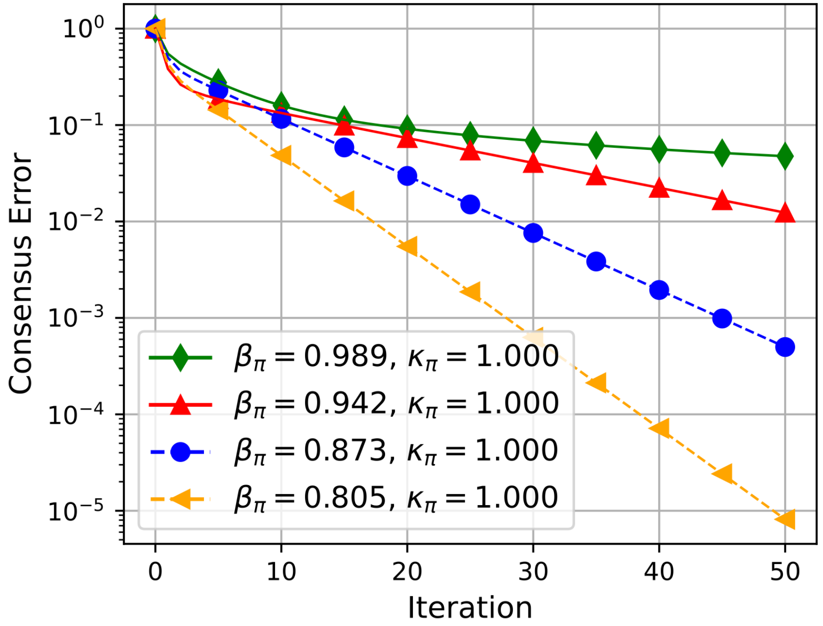


- β_π and κ_π is orthogonal
- The influence of κ_π can be highly non-trivial !

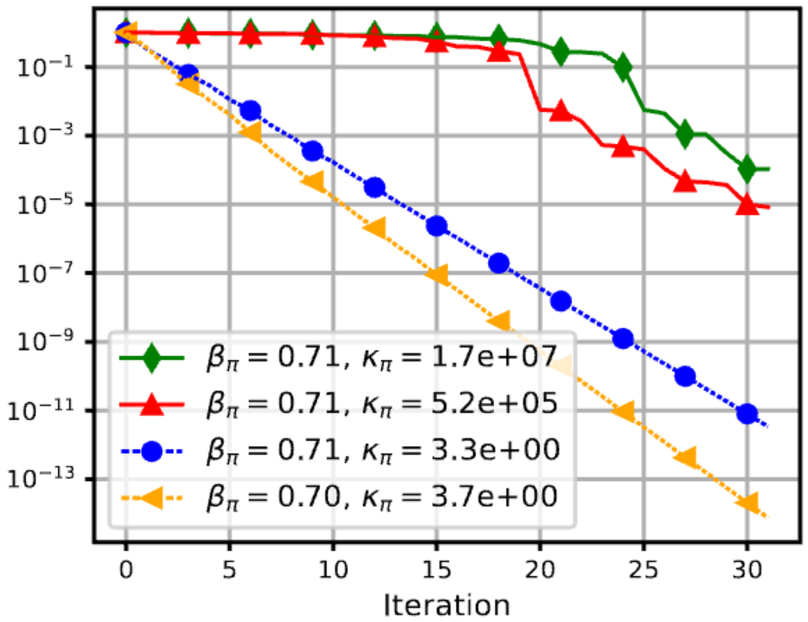
Push-sum decentralized averaging: simulation

- Metrics β_π and κ_π together precisely reflects the influence of network on push-sum averaging

$\kappa_\pi = \text{const. and } \beta_\pi \to 1$



$\beta_\pi = \text{const. and } \kappa_\pi \to \infty$



- Both metrics are indispensable

Push-sum gradient tracking

- Recall the distributed stochastic optimization problem

$$\min_{x \in \mathbb{R}^d} \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \quad \text{where} \quad f_i(x) = \mathbb{E}_{\xi_i \sim D_i} F(x; \xi_i).$$

- Given a column-stochastic matrix W, the push-sum gradient tracking [NOS17] is

$$\begin{aligned}
\mathbf{x}^{(k+1)} &= W(\mathbf{x}^{(k)} - \gamma \mathbf{y}^{(k)}) \\
v^{(k+1)} &= W v^{(k)} \\
V^{(k+1)} &= \mathrm{diag}(v^{(k+1)}) \\
\mathbf{w}^{(k+1)} &= V^{(k+1)^{-1}} \mathbf{x}^{(k+1)} \\
\mathbf{y}^{(k+1)} &= W(\mathbf{y}^{(k)} + \nabla F(\mathbf{w}^{(k+1)}; \boldsymbol{\xi}^{(k+1)}) - \nabla F(\mathbf{w}^{(k)}; \boldsymbol{\xi}^{(k)}))
\end{aligned}$$

Push-sum gradient tracking: convergence

Theorem 3

Suppose each $f_i(x)$ is L-smooth, each local stochastic gradient $\nabla F(x;\xi_i)$ is unbiased and has bounded variance. Given a column-stochastic weight matrix W, the convergence rate of Push-DIGing satisfies

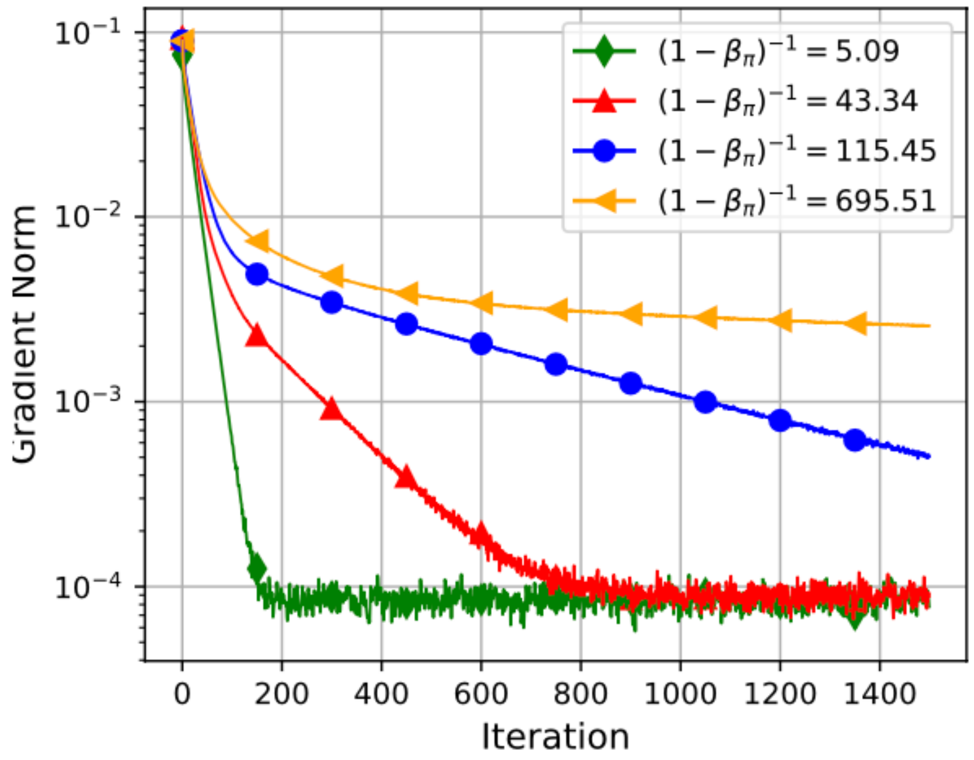
$$\frac{1}{K}\sum_{i=0}^{K-1}\mathbb{E}[\|\nabla f(\bar{x}^{(k)}\|_2^2] \lesssim \frac{\sigma}{\sqrt{nK}} + \frac{\beta_\pi^{\frac{2}{3}}\kappa_\pi^{\frac{5}{3}}\sigma^{\frac{2}{3}}}{(1-\beta_\pi)K^{\frac{2}{3}}} + \frac{\beta_\pi\kappa_\pi^3(1+\kappa_\pi\beta_\pi)}{(1-\beta_\pi)^2K} + \frac{1}{K}.$$

The **first** rate that clarifies the influence of digraphs on decentralized algorithms

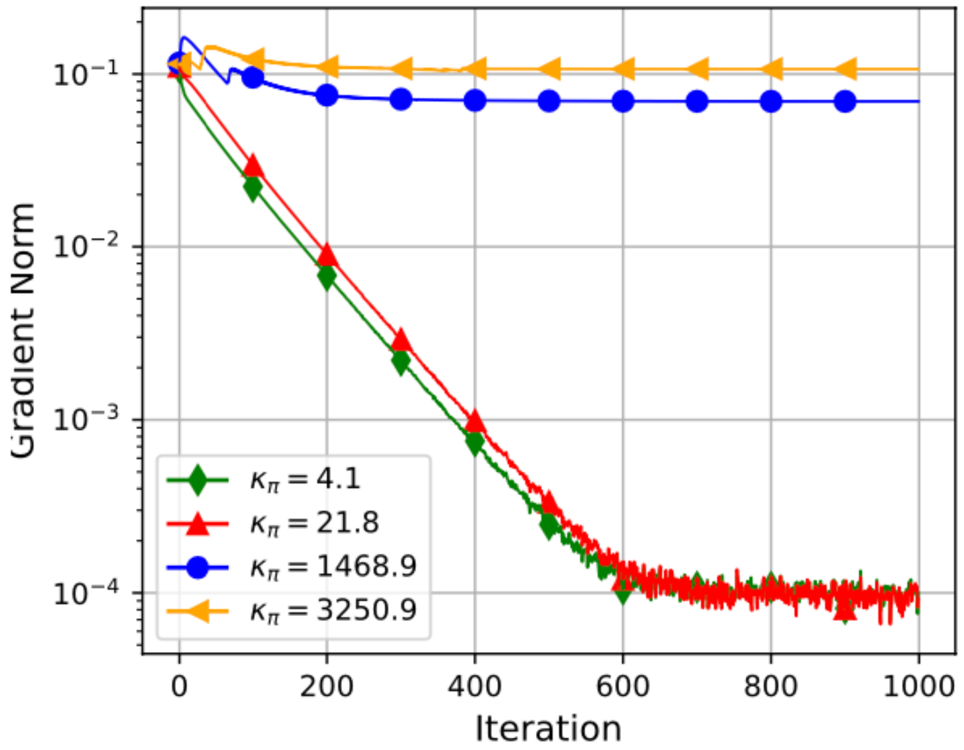
Push-sum gradient tracking: simulation

- Logistic regression with non-convex regularization terms

$\kappa_\pi = \text{const. and } \beta_\pi \to 1$



$\beta_\pi = \text{const. and } \kappa_\pi \to \infty$



Push-sum gradient tracking: comparison with existing works

Algorithm	Rate (A.)	Rate (F.T.)	Transient Stage
Gradient-Push [4]	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	N.A.	N.A.
Push-DIGing [15]	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	N.A.	N.A.
Push-DIGing (Ours)	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}} + \frac{\beta_\pi \kappa_\pi^3 (1+\kappa_\pi \beta_\pi)}{(1-\beta_\pi)^2 K}$	$\frac{n\kappa_\pi^8}{(1-\beta_\pi)^4}$

The influence of κ_π is substantial, especially when $\kappa_\pi = O(2^n)$

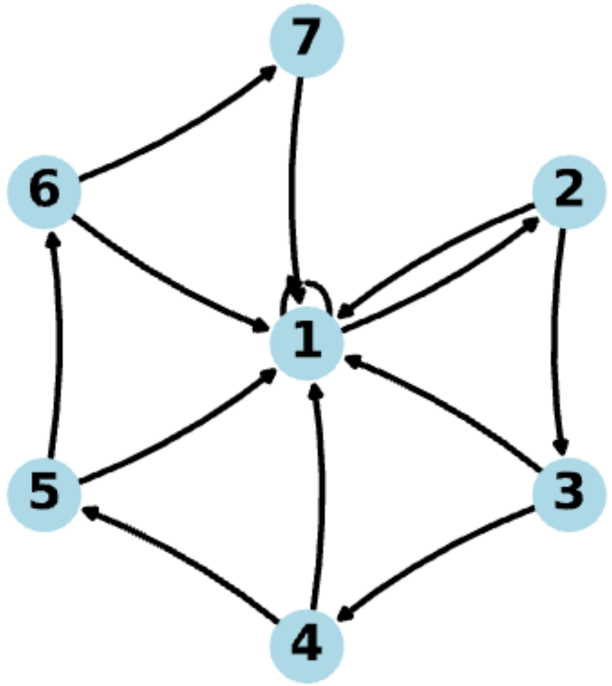
What is the **best-possible dependence** on κ_π and β_π ?

北京大学 PEKING UNIVERSITY

Part 05

Lower bound over digraphs

A challenging digraph



For this graph, we prove that

$$\pi \propto \left(2^{n-1}, 2^{n-2}, \ldots, 1\right)^\top \text{ and } \kappa_\pi = 2^{n-1}$$

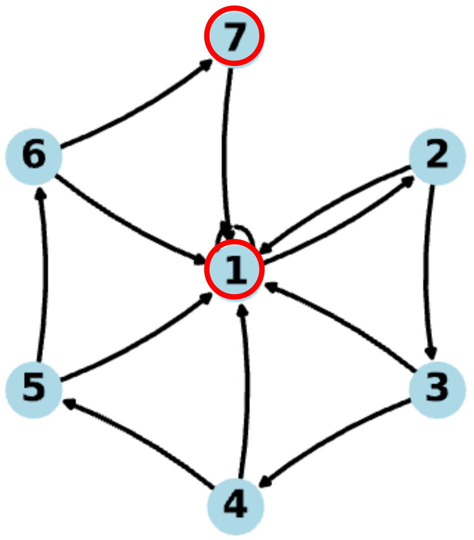
Also, the spectral gap is constant

$$\beta_\pi = 1/\sqrt{2} \quad \forall n$$

Lower bound of communication rounds: core idea

- For non-convex centralized deterministic optimization, the lower bound rate is $\Omega(L\Delta/K)$

- For the digraph below, each effective message passing requires n hops.



require n hops per iteration

Communication lower bound

$$\Omega(\frac{nL\Delta}{K})$$

$$n = \frac{1+\ln(\kappa_\pi)}{1-\beta_\pi}$$

Communication lower bound

$$\Omega(\frac{(1+\ln(\kappa_\pi))L\Delta}{1-\beta_\pi})$$

Convergence lower bound over digraphs

Theorem 4

For any given $L \geq 0$, $n \geq 2$, $\sigma \geq 0$, and $\tilde{\beta} \in [1/\sqrt{2}, 1 - 1/n]$, there exists a set of loss functions $\{f_i\}_{i=1}^n \in \mathcal{F}_{\Delta,L}$, a set of stochastic gradient oracles in $\mathcal{O}_{\sigma^2}$, and a column-stochastic matrix $W \in \mathbb{R}^{n\times n}$ with $\beta_\pi = \tilde{\beta}$ and $\ln(\kappa_\pi) = \Omega(n(1-\beta_\pi))$, such that the convergence of any $A \in \mathcal{A}_W$ starting form $x_i^{(0)} = x^{(0)}, \forall\, 1 \leq i \leq n$ with K iterations is lower bounded by

$$\mathbb{E}[\|\nabla f(x^{(K)})\|_2^2] = \Omega\left(\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}} + \frac{(1+\ln(\kappa_\pi))L\Delta}{(1-\beta_\pi)K}\right).$$

Big gap between lower and upper bound

北京大学 PEKING UNIVERSITY

Algorithm	Rate (A.)	Rate (F.T.)	Transient Stage
Gradient-Push [4]	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	N.A.	N.A.
Push-DIGing [15]	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	N.A.	N.A.
Push-DIGing (Ours)	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}} + \boxed{\frac{\beta_\pi\kappa_\pi^3(1+\kappa_\pi\beta_\pi)}{(1-\beta_\pi)^2K}}$	$\frac{n\kappa_\pi^8}{(1-\beta_\pi)^4}$
Lower Bound (Ours)	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}} + \boxed{\frac{(1+\ln(\kappa_\pi))L\Delta}{(1-\beta_\pi)K}}$	$\frac{n(1+\ln(\kappa_\pi))^2}{(1-\beta_\pi)^2}$

Part 06

Optimal decentralized algorithm over digraphs

Recall push-sum gradient tracking

$$\mathbf{x}^{(k+1)} = W(\mathbf{x}^{(k)} - \gamma \mathbf{y}^{(k)})$$

$$v^{(k+1)} = W v^{(k)}$$

$$V^{(k+1)} = \mathrm{diag}(v^{(k+1)})$$

$$\mathbf{w}^{(k+1)} = {V^{(k+1)}}^{-1} \mathbf{x}^{(k+1)}$$

$$\mathbf{y}^{(k+1)} = W(\mathbf{y}^{(k)} + \nabla F(\mathbf{w}^{(k+1)}; \boldsymbol{\xi}^{(k+1)}) - \nabla F(\mathbf{w}^{(k)}; \boldsymbol{\xi}^{(k)}))$$

Multiple-Gossip Push-sum Gradient tracking

Improve it with **multiple-gossip** and **mini-batch gradient**

$$
\begin{aligned}
\mathbf{x}^{(k+1)} &= W^R(\mathbf{x}^{(k)} - \gamma \mathbf{y}^{(k)}) \\
v^{(k+1)} &= W^R v^{(k)} \\
V^{(k+1)} &= \mathrm{diag}(v^{(k+1)}) \\
\mathbf{w}^{(k+1)} &= {V^{(k+1)}}^{-1} \mathbf{x}^{(k+1)} \\
\mathbf{y}^{(k+1)} &= W^R(\mathbf{y}^{(k)} + g_R^{(k+1)} - g_R^{(k)})
\end{aligned}
$$

where $g_R^{(k)} = \frac{1}{R}\sum_{r=1}^{R} \nabla F(\mathbf{w}^{(k)}; \xi^{(k,r)})$ is the mini-batch stochastic gradient

Multiple-Gossip Push-sum Gradient tracking

北京大學 PEKING UNIVERSITY

Theorem 5

Suppose each $f_i(x)$ is L-smooth, each local stochastic gradient $\nabla F(x;\xi_i)$ is unbiased and has bounded variance, and the weight matrix W is column-stochastic, by setting $R = \frac{(1+\sqrt{\ln(\kappa_\pi)})^2}{1-\beta_\pi}$, $T = KR$, the convergence of MG-Push-DIGing satisfies

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|\nabla f(\bar{x}^{(k)})\|_2^2] = \widetilde{\mathcal{O}}\left(\frac{\sigma\sqrt{L\Delta}}{\sqrt{nT}} + \frac{(1+\ln(\kappa_\pi))L\Delta}{(1-\beta_\pi)T}\right),$$

where $\widetilde{\mathcal{O}}(\cdot)$ absorbs logarithmic factors independent of κ_π and β_π.

Lower bound and upper bound are nearly-matched

PEKING UNIVERSITY 北京大学

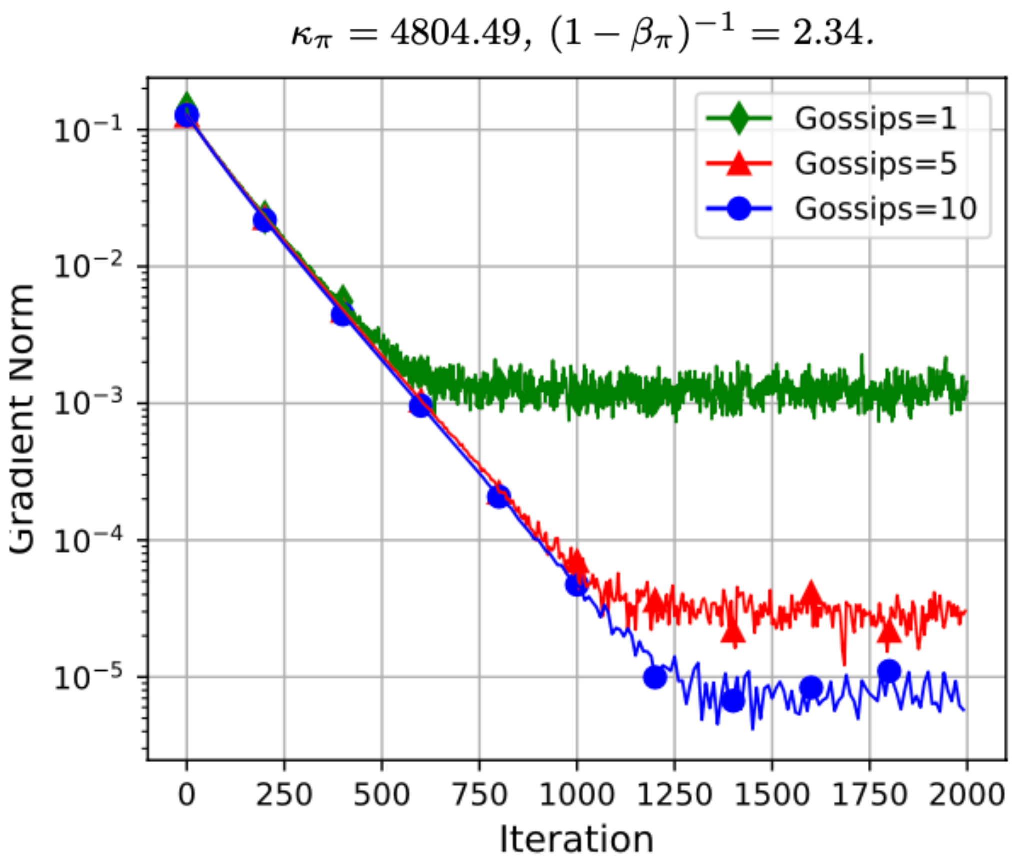
Algorithm	Rate (A.)	Rate (F.T.)	Transient Stage
Gradient-Push [4]	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	N.A.	N.A.
Push-DIGing [15]	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	N.A.	N.A.
Push-DIGing (Ours)	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}} + \frac{\beta_\pi\kappa_\pi^3(1+\kappa_\pi\beta_\pi)}{(1-\beta_\pi)^2K}$	$\frac{n\kappa_\pi^8}{(1-\beta_\pi)^4}$
MG-Push-DIGing (Ours)	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	$\boxed{\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}} + \frac{(1+\ln(\kappa_\pi))L\Delta}{(1-\beta_\pi)K}}$	$\frac{n(1+\ln(\kappa_\pi))^2}{(1-\beta_\pi)^2}$
Lower Bound (Ours)	$\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}}$	$\boxed{\frac{\sigma\sqrt{L\Delta}}{\sqrt{nK}} + \frac{(1+\ln(\kappa_\pi))L\Delta}{(1-\beta_\pi)K}}$	$\frac{n(1+\ln(\kappa_\pi))^2}{(1-\beta_\pi)^2}$

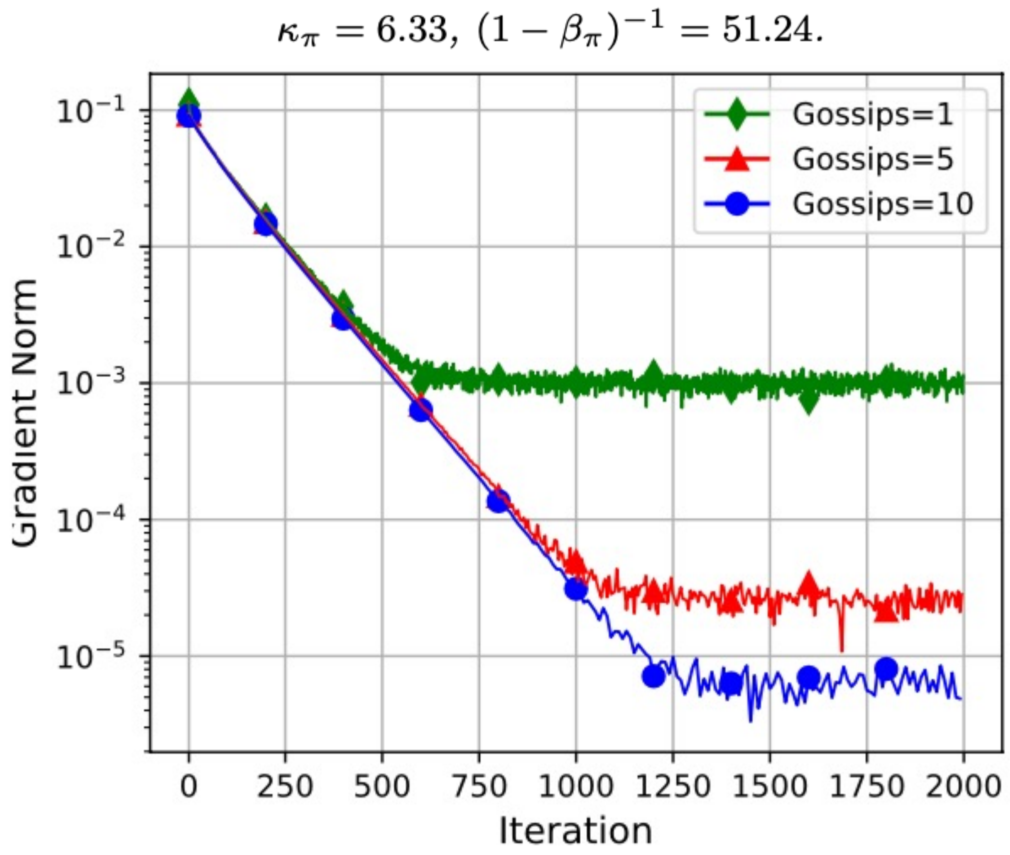
Our lower bound is nearly-tight

Our developed MG-Push-DIGing algorithm is nearly optimal

Simulations

- Logistic regression with non-convex regularizer. Multiple Gossip improves both κ_π and $(1-\beta_\pi)^{-1}$





Summary

- The influence of diagraphs on decentralized algorithms is known in previous work
- We identify two effective metrics (spectral gap and equilibrium skewness) that can jointly capture the influence of digraphs
- The two metrics are orthogonal to each other; both of them are indispensable
- We establish the lower bound and develop a nearly-optimal algorithm to attain the lower bound

Future work

- Clarify the influence of row-stochastic digraphs on decentralized algorithms

  [Ongoing]

- Clarify the influence of digraphs over push-pull algorithms

  [Almost done]

- Lower bound and optimal algorithms for pull-sum and push-sum family

  [Very challenging]

Thank you!

Liyuan Liang, Xinmeng Huang, Ran Xin, and Kun Yuan, “*Towards Better Understanding the Influence of Directed Networks on Decentralized Stochastic Optimization*”, arXiv:2312.04928, 2023