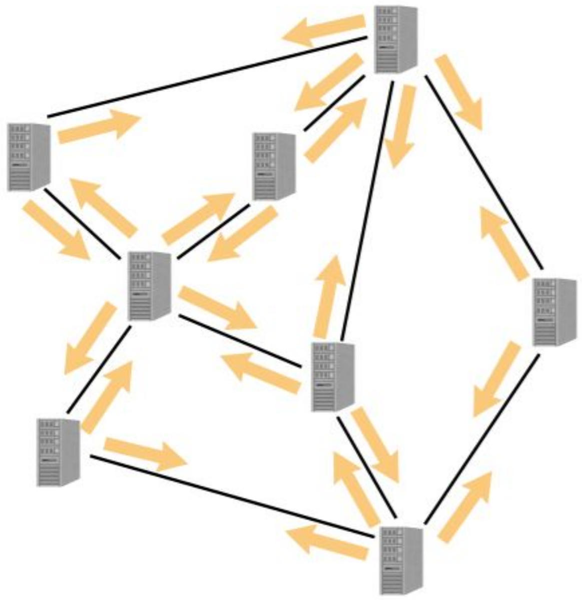
GNN v.s. Distributed system

Zixiang Chen, Jiafan He, Weitong Zhang

Background: distributed networks

1. Each user is connected to some neighbors
2. Each user can only communicate with neighbors.

Locality connection -> Global result



Background: distributed algorithms

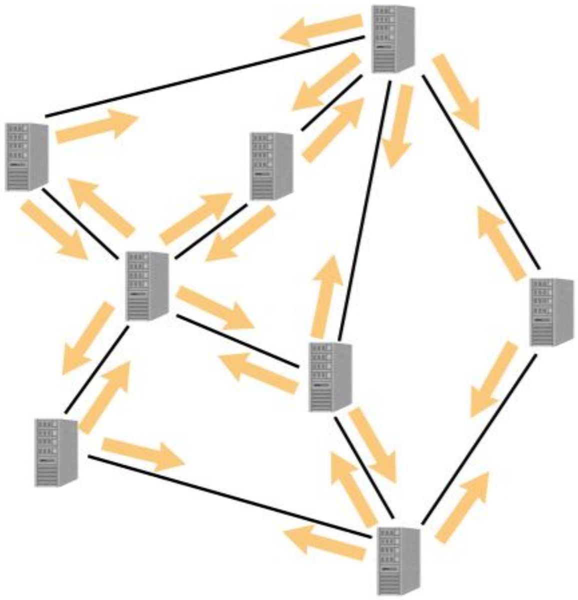
1. In each communication round, every user runs the same program and send a message to its neighbor.
2. After receive the messages, every user computes the next messages and next state based on these messages.

Network <-> Graph

User <-> Node

Connection <-> Edge

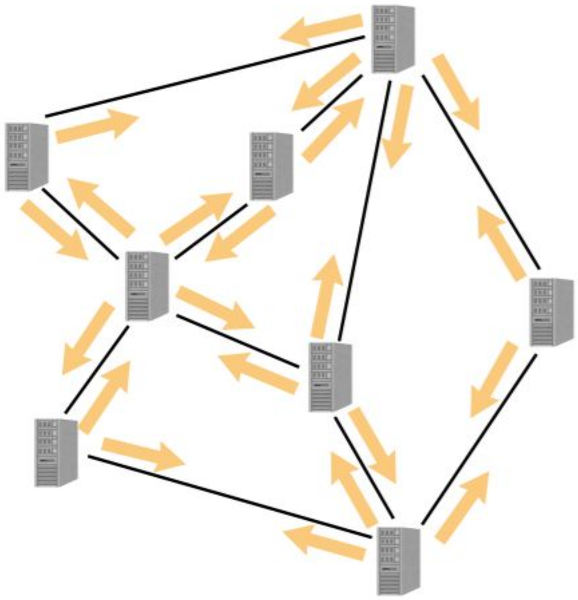
State <-> Parameter



Background: distributed local algorithms

Local algorithm: After a constant number of rounds, each node outputs an answer based on the states and received messages

Constant round <-> Constant layers in GNN



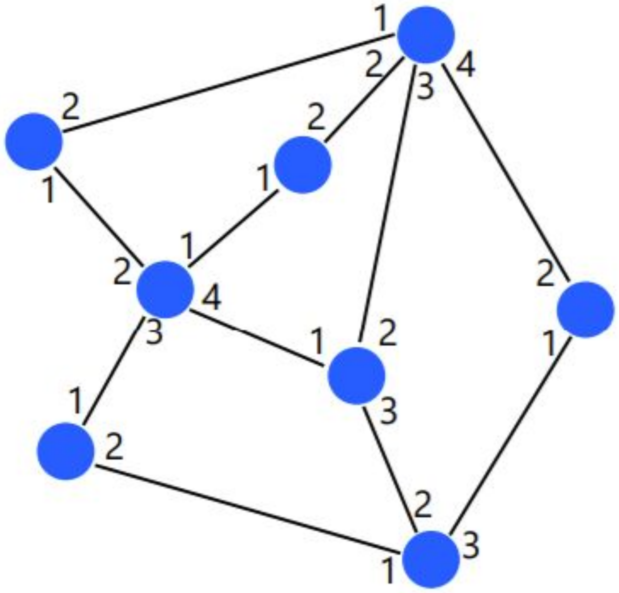
Background: Communication assumption

1. Vector-vector consistent model: VV(1)

1:Each channel has a different number

2:Each user can send different message to different neighbor

(Each user knows the channel that a message come from)



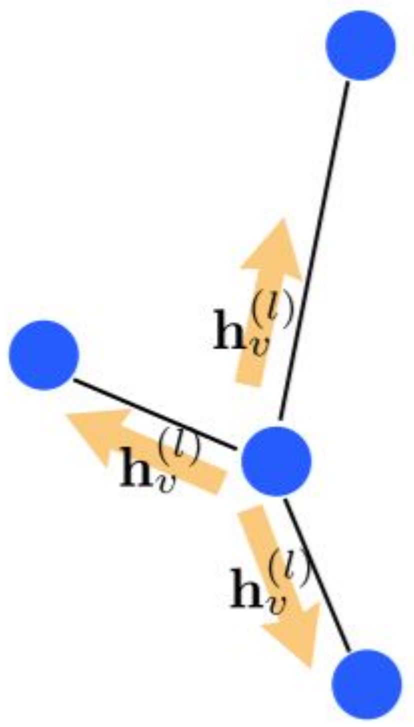
(a) A graph with a port numbering.

Background: Communication assumption

2. Multiset-broadcasting model: MB(1)

1:Each channel doesn’t have a different number

2:Each user send the same message to different neighbor



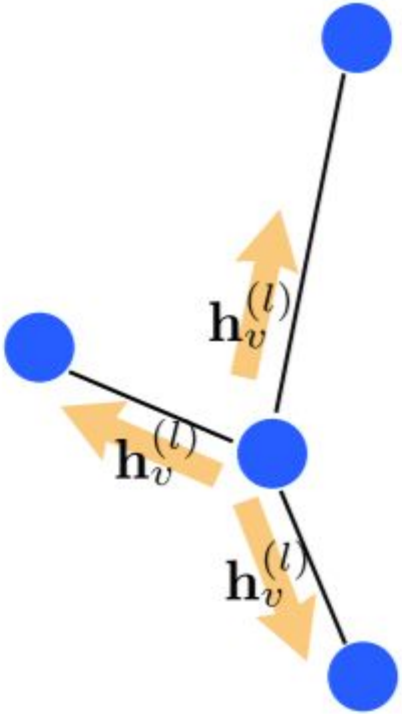
Background: Communication assumption

3. Set-broadcasting: SB(1)

1:Each channel doesn’t have a different number

2:Each user send the same message to different neighbor

3:Each node cannot count the number of repeating messages



GNN on distributed systems: MB-GNN (Sato et. al, 2019)

- Multiset-Broadcasting GNN (MB-GNN)
- Each vertex u broadcast h_u, received in a multiset
- Aggregate function
- Update function
- Same as vanilla GNN function, GCN, etc.

$$\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v \qquad (\forall v \in V),$$

$$\boldsymbol{a}_v^{(k)} = f_{\text{aggregate}}^{(k)}(\{\!\{\boldsymbol{h}_u^{(k-1)} \mid u \in \mathcal{N}(v)\}\!\}) \qquad (\forall k \in [L], v \in V),$$

$$\boldsymbol{h}_v^{(k)} = f_{\text{update}}^{(k)}(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{a}_v^{(k)}) \qquad (\forall k \in [L], v \in V),$$

GNN on distributed systems:

- MB-GNN: Message record in a set (unique elements)

$$\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v \qquad (\forall v \in V),$$

$$\boldsymbol{a}_v^{(k)} = f_{\text{aggregate}}^{(k)}(\{\boldsymbol{h}_u^{(k-1)} \mid u \in \mathcal{N}(v)\}) \qquad (\forall k \in [L], v \in V),$$

$$\boldsymbol{h}_v^{(k)} = f_{\text{update}}^{(k)}(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{a}_v^{(k)}) \qquad (\forall k \in [L], v \in V),$$

- VV-GNN: vertex sends different message to different neighbor

$$\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v \qquad (\forall v \in V),$$

$$\boldsymbol{a}_v^{(k)} = f_{\text{aggregate}}^{(k)}(\{(p(v,u), p(u,v), \boldsymbol{h}_u^{(k-1)}) \mid u \in \mathcal{N}(v)\}) \qquad (\forall k \in [L], v \in V),$$

$$\boldsymbol{h}_v^{(k)} = f_{\text{update}}^{(k)}(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{a}_v^{(k)}) \qquad (\forall k \in [L], v \in V),$$

GNN v.s. Distributed local algorithms:

Theorem:

- MB-GNN algorithm = MB(1) algorithm
- SB-GNN algorithm = SB(1) algorithm
- VV-GNN algorithm = VV(1) algorithm

What is equal?

- For any function f in MB-GNN, there exists an MB(1) algorithm to get the same results

CPNGNN: parameterize the functions

CPNGNNs. CPNGNNs concatenate neighboring embeddings in the order of the port numbering.

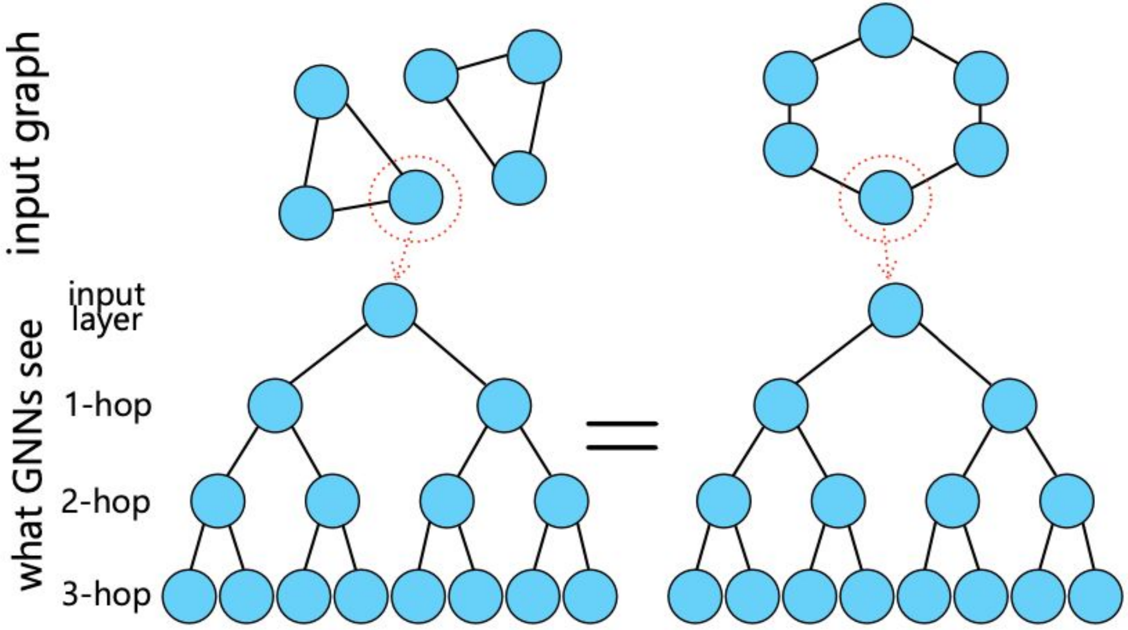
$$\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v \qquad (\forall v \in V),$$

$$\boldsymbol{a}_v^{(k)} = \boldsymbol{W}^{(k)}[\boldsymbol{h}_v^{(k-1)\top}, \boldsymbol{h}_{u_{v,1}}^{(k-1)\top}, p(u_{v,1}v), \ldots, \boldsymbol{h}_{u_{v,\Delta}}^{(k-1)\top}, p(u_{v,\Delta}, v)]^\top \qquad (\forall k \in [L], v \in V),$$

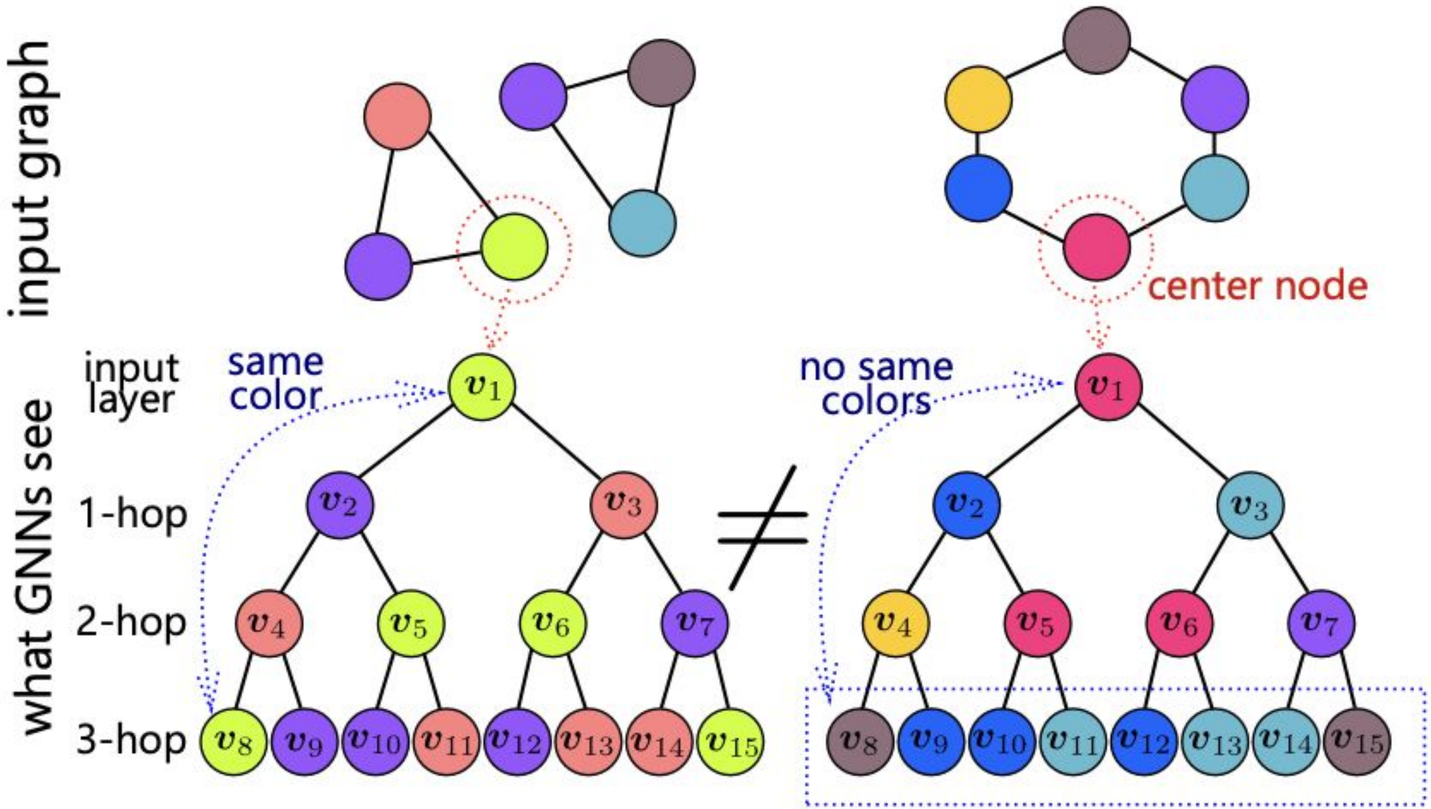
$$\boldsymbol{h}_v^{(k)} = \mathrm{ReLU}(\boldsymbol{a}_v^{(k)}) \qquad (\forall k \in [L], v \in V),$$

- Use neural networks parameterization
- Can approximate the MB-GNN if the number of neighborhood is bounded
- Also discuss the time complexity of CPNGNN, including some polynomial-time approximation solution for NP-hard problems.

Random Feature improves GNN on distributed systems



(a) Identical Features.



(b) Random Features.

Definition

$$R(G, v, L) = (\mathcal{N}_L(v), \{\{x, y\} \in E \mid x, y \in \mathcal{N}_L(v)\})$$

$$G = (V, E, \boldsymbol{X}), \text{ where } \boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n \times d_I}$$

support $\mathcal{C}$ of the feature vectors is finite (i.e., $|\mathcal{C}| < \infty$).

Definition 27 (Isomorphism with a center node)**.** Let $G = (V, E, \boldsymbol{X})$ and $G' = (V', E', \boldsymbol{X}')$ be graphs and $v \in V$ and $v' \in V'$ be nodes. (G, v) and (G', v') are isomorphic if there exists a bijection $f \colon V \to V'$ such that $(x, y) \in E \Leftrightarrow (f(x), f(y)) \in E'$, $\boldsymbol{x}_x = \boldsymbol{x}'_{f(x)}$ $(\forall x \in V)$, and $f(v) = f(v')$. $(G, v) \simeq (G', v')$ denotes (G, v) and (G', v') are isomorphic.

Algorithm

Algorithm 1 rGINs: GINs with random features

Require: $G = (V, E, \boldsymbol{X})$, Distribution μ, Parameters $\boldsymbol{\theta}$.

Ensure: Embeddings $[\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n]^\top \in \mathbb{R}^{n \times d_O}$

Assign random features $\boldsymbol{r}_v \sim \mu \quad (\forall v \in V)$

return $\mathrm{GIN}_{\boldsymbol{\theta}}((V, E, \mathrm{CONCAT}([\boldsymbol{X}, [\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n]^\top])))$.

Theorem 28 (Sato et al. (2020), Theorem 1). *$\forall L \in \mathbb{Z}^+, \Delta \in \mathcal{Z}^+$, for any finite feature space $\mathcal{C}$ ($|\mathcal{C}| < \infty$), for any set $\mathcal{G} = \{(G, v)\}$ of pairs of a graph $G = (V, E, \boldsymbol{X})$ and a center node $v \in V$ such that the maximum degree of G is at most Δ and $\boldsymbol{x}_u \in \mathcal{C}$ ($\forall u \in V$), there exists $q \in \mathbb{R}^+$ such that for any discrete distribution μ with finite support X such that $\mu(x) \leq q$ ($\forall x \in X$), there exists a set of parameters $\boldsymbol{\theta}$ such that for any pair of a graph $G = (V, E, \boldsymbol{X})$ and a center node $v \in V$ such that the maximum degree of G is at most Δ and $\boldsymbol{x}_u \in \mathcal{C}$ ($\forall u \in V$)*

- *if $\exists (G', v') \in \mathcal{G}$ such that $(G', v') \simeq (R(G, v, L), v)$ holds, $\mathrm{rGIN}(G, \mu, \boldsymbol{\theta})_v > 0.5$ holds with high probability.*
- *if $\forall (G', v') \in \mathcal{G}$, $(G', v') \not\simeq (R(G, v, L), v)$ holds, $\mathrm{rGIN}(G, \mu, \boldsymbol{\theta})_v < 0.5$ holds with high probability.*

Sato, Ryoma, Makoto Yamada, and Hisashi Kashima. "Random features strengthen graph neural networks." *arXiv preprint arXiv:2002.03155* (2020).

Conclusion

- GNN is similar with distributed algorithms
- Random Feature can improve the performance of GNN
- Like the GNN and WL test, one can compare the result with GNN and distributed algorithms in different setting.

Thank you!