On the Expressive Power of Graph Neural Network

Zixiang Chen[*†] and Weitong Zhang[*‡] and Jiafan He[*§]

Abstract

We investigate the expressive power issue of the Graph Neural Network (GNN) by comparing the GNN structure with distributed systems and local algorithms. Following Sato (2020) and a bunch of related papers, we show the connection between GNN and local algorithms.

1 Introduction

Graph Neural Networks (GNN)(Gori et al., 2005; Scarselli et al., 2008) are powerful machine learning models for various graph learning problems, which learn node representation with recurrent neural networks. The application includes the recommendation system (Ying et al., 2018), question-answering systems (Schlichtkrull et al., 2018) and combinatorial problems (Dai et al., 2017).

However, the empirical success of GNN hasn't been well understood. One of the most important issues is the expressive power of GNNs. The expressive ability of GNNs may not be as powerful as we have expected. To be specific, some pairs of graphs may not be distinguished by the GNNs (Xu et al., 2018), many combinatorial problems also cannot be solved by GNNs because GNNs are at most as powerful as distributed local algorithms (Angluin, 1980; Suomela, 2013).

By providing a comprehensive overview of the expressive power of GNNs, this survey may help readers to understand the current limitation of GNNs and inspire the improvement of the GNNs model with more expressive architectures.

2 Local Algorithms on Distributed Systems

In this section, we introduce the background of distributed local algorithm. In a distributed network, there exist a series of different processors and each processor is only directly connected to part of all processors. Due to the locality of inter-connections, each processor can only communicate with its' neighbors through the channel and the goal of distributed network is deciding a global result with performing some computation on each individual processor. In particular, we model the distributed network as an un-directed graph, where each node corresponds the processor and each edge represents the interconnection between two processors. Figure 1 shows an example of the distributed network.

For a distributed algorithm, each processor runs the same program and sends a message to its neighbor at the same time in each communication round. After receiving neighbors' messages, every

[*]Department of Computer Science, University of California, Los Angeles, CA 90095, USA
[†]Email: `chenzx19@cs.ucla.edu`; UID: 205348320
[‡]Email: `wt.zhang@ucla.edu`; UID: 705302329
[§]Email: `jiafanhe19@ucla.edu`; UID: 405351222

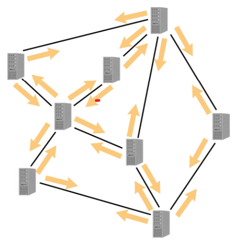

Figure 1: Distributed network with 8 nodes and 11 edges

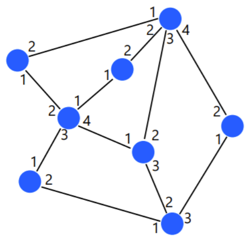

Figure 2: Vector-vector consistent model

processor computes the next messages and next state based on these messages and the current state. For a distributed algorithm, if it can solves a problem on distributed network within constant communication rounds, it is called distributed local algorithm. Distributed local algorithms were first studied by Angluin (1980); Linial (1992); Naor and Stockmeyer (1995) and it can be divided into several categories based on the assumption of communication.

1:Vector-vector consistent model: VV(1) (Suomela, 2013) In Vector-vector consistent model, each node has several ports and each edge incident to this node is connected to one of the ports. Figure 2 shows an example of the Vector-vector consistent model. In each communication rounds, every processor can send a different message to a different neighbor and each node also knows the port number that the neighboring node submits the message to.

2:Multiset-broadcasting model: MB(1) (Hella et al., 2015) For Multiset-broadcasting model, each channel doesn't have a different port number and each processor should send the same message to different neighbors.

3:Set-broadcasting: SB(1) (Hella et al., 2015) For a Multiset-broadcasting model, if each node receives messages as a set and cannot count the number of repeating messages, this model is called Set-broadcasting model.

3 GNN and Local Algorithms

3.1 GNN formulation

Equipped with the knowledge on the distributed systems and local algorithms. Generally speaking, each layer of the GNN should contain these two parts:

- Aggregation: aggregate the received information vector $\mathbf{h}$ from other vertices together and generate the feature vector $\mathbf{a}$ by aggregation function f_a.
- Update: update the contextual vector $\mathbf{x}$ using the aggregation vector $\mathbf{a}$ by update function f_u

We could formulate the GNN structure for different local algorithms:

MB-GNN The MB-GNN is formulated from the multi-set broadcast (MB) setting as mentioned above. In this setting, each vertex u could broadcast its information vector $\mathbf{h}_u$ to all of its neighborhoods. The received vectors are kept in a multi-set, which allows duplicate vectors from different vertices. The GNN should be formulated as

$$\begin{aligned} \mathbf{h}_v &= \mathbf{x}_v^{(0)} && (\forall v \in V), \\ \mathbf{a}_v^{(k)} &= f_a^{(k)}(\{\{\mathbf{h}_u^{(k-1)} | u \in \mathcal{N}(v)\}\}) && (\forall k \in [L], v \in V), \\ \mathbf{h}_v^{(k)} &= f_u^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)}) && (\forall k \in [L], v \in V), \end{aligned}$$

where L is the total number of layers in GNN. On can find out that MB-GNN has the same setting with the vanilla GNN, which also shows a strong relationship between GNN and those local algorithms.

SB-GNN The SB-GNN is following the set broadcast setting, as mentioned above. Since the received vectors are kept in a set instead of a multi-set, there is no duplicate vectors and each vertex only keeps a unique version of the vectors sent from its neighborhood. Thus the GNN should be formulated as

$$\begin{aligned} \mathbf{h}_v &= \mathbf{x}_v^{(0)} && (\forall v \in V), \\ \mathbf{a}_v^{(k)} &= f_a^{(k)}(\{\mathbf{h}_u^{(k-1)} | u \in \mathcal{N}(v)\}) && (\forall k \in [L], v \in V), \\ \mathbf{h}_v^{(k)} &= f_u^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)}) && (\forall k \in [L], v \in V), \end{aligned}$$

where L is the total number of layers in GNN. We can found that the MB-GNN and SB-GNN looks similar except for changing the multi-set to set.

VV-GNN Under the Vector-Vector (VV) consistent model, since each vertex could send different to its different neighborhood, we need to add information between different vertices. Thus the GNN is formulated as

$$\begin{aligned} \mathbf{h}_v &= \mathbf{x}_v^{(0)} && (\forall v \in V), \\ \mathbf{a}_v^{(k)} &= f_a^{(k)}(\{p(u,v), p(v,u), \mathbf{h}_u^{(k-1)} | u \in \mathcal{N}(v)\}) && (\forall k \in [L], v \in V), \\ \mathbf{h}_v^{(k)} &= f_u^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)}) && (\forall k \in [L], v \in V), \end{aligned}$$

Under these formulation of GNN, Sato et al. (2019) has proved the following theorem to describe the relationship between the GNN and its corresponding local algorithms.

Theorem 3.1. Let $\mathcal{L}$ be MB, SB, or VV. For any algorithm $\mathcal{A}$ of the $\mathcal{L}(1)$ model, there exists a $\mathcal{L}$-GNN that the output is same as $\mathcal{A}$. For any $\mathcal{L}$-GNN $\mathcal{N}$, there exists an algorithm $\mathcal{A}$ of the $\mathcal{L}(1)$ model that the output is the same as the embedding computed by $\mathcal{N}$.

Theorem 3.1 shows the equivalence of GNN and its corresponding local algorithms. The intuition behind this proof is that we can choose arbitrary aggregation function f_a and update function f_u. Furthermore, since Hella et al. (2015) has shown the following lemma to show the relationship of MB, SB and VV local algorithm.

Theorem 3.2. The class of the functions that the VV(1) model can compute is strictly wider than the class of the functions that the MB(1) model can compute, and the class of the functions that the MB(1) model can compute is strictly wider than the class of the functions that the SB(1) model can compute.

Thus we can easily get the relationship between MB-GNN, SB-GNN and VV-GNN as proposed in Xu et al. (2018); Sato (2020)

Corollary 3.3. The class of the functions that the VV-GNNs model can compute is strictly wider than the class of the functions that the MB-GNNs model can compute, and the class of the functions that the MB-GNNs model can compute is strictly wider than the class of the functions that the SB-GNNs model can compute.

3.2 Parameterizing GNN functions

Since in previous section we have been focusing on the **arbitrary** aggregate function and update function, it would be natural to ask whether we can use the parameterization like Multi-Layer-Perception to craft the GNN. To tackle this question, Sato et al. (2019) proposed CPNGNN to concatenate neighboring embedding in the order of the port numbering, which is defined as

$$\begin{aligned}
\mathbf{h}_v^{(0)} &= \mathbf{x}_b && (\forall v \in V),\\
\mathbf{a}_v^{(k)} &= \mathbf{W}^{(k)}[\mathbf{h}_v^{(k-1)\top}, \mathbf{h}_{u_{v,1}}^{(k-1)\top}, p(u_{v,1}, v), \cdots, \mathbf{h}_{u_{v,\Delta}}^{(k-1)\top}, p(u_{v,\Delta}, v)]^\top && (\forall k \in L, v \in V),\\
\mathbf{h}_v^{(k)} &= \sigma(\mathbf{a}_v^{(k)}) && (\forall k \in L, v \in V),
\end{aligned}$$

where $\sigma(x) = \max\{x, 0\}$ is the well-known ReLU function. Sato et al. (2019) has shown that if the maximum degree of input graph is controlled by Δ, CPNGNN would be most powerful among VV-GNNs

Theorem 3.4. If the degrees of the nodes are bounded by a constant and the size of the support of node features is finite, for any VVC-GNNs $\mathcal{N}$, there exist parameters of CPNGNNs such that for any (bounded degree) graph $G = (V, E, \mathbf{X})$, the embedding computed by the CPNGNN is arbitrary close to the embedding computed by $\mathcal{N}$.

Based on those claims, Sato et al. (2019) finally shows that CPNGNN could be used to solve several combinations problem by converting the corresponding local algorithm to GNNs. They also shows the polynominal time using CPNGNN to solve those problem. We refer the readers to check Sato (2020); Sato et al. (2019) for the detailed time complexity issue.

4 Random Features strengthen the GNN

A recent line of works (Sato et al., 2020; Corso et al., 2020) adds random features to each node enables GNNs to distinguish a wider class of graphs and model more efficient algorithms. These

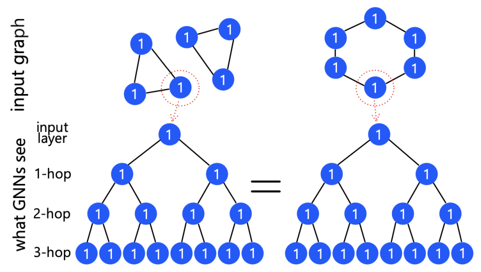

Figure 3: Identical Feature

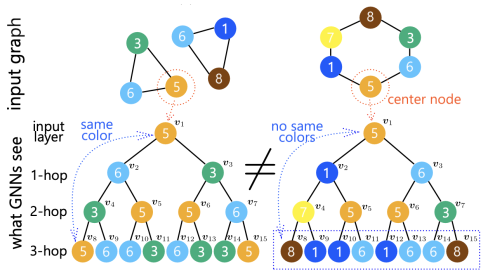

Figure 4: Random Feature

algorithms are based on either local orderings or unique global identifiers and make GNNs universal and powerful. Similarly, Dasoulas et al. (2019) use the random coloring method to identify the nodes uniquely.

4.1 Why random feature works

Here we use the examples 3 and 4 in Sato et al. (2020) to illustrate the advantage of random feature. It is known to all that message passing GNNs cannot know the entire input graph. What it can know is actually the breadth-first search tree.

Figure 3 shows that the the breadth first search tree generated by a center node in two cycles of three nodes is the same with the one generated by a center node in a cycle of six node. Therefore GNNs with identical features (such as degree features) cannot distinguish a node in a cycle of three nodes with a node in a cycle of six node. Figure 4 shows the power of random feature. We assign the random features to all nodes and then use a toy model that concatenates features of all nodes. $\mathbf{v}_1$ is the first dimension of the embedding v which is the random feature of the center node. $\mathbf{v}_2, \mathbf{v}_3$ are the random features of the one-hop nodes. The $\mathbf{v}_4, \ldots, \mathbf{v}_7$ are the random features of the two-hop nodes. $\mathbf{v}_8, \ldots, \mathbf{v}_{15}$ are the random features of the three-hop nodes. Therefore we can check whether the center node is involved in a 3-cycle by checking the embedding on the union of certain hyper planes

$$\{\mathbf{v}|\mathbf{v}_1 = \mathbf{v}_8\} \cup \{\mathbf{v}|\mathbf{v}_1 = \mathbf{v}_9\} \cup \ldots \cup \{\mathbf{v}|\mathbf{v}_1 = \mathbf{v}_{15}\}.$$

It is also worth noting that what really matters is the relationship between the values rather than the value itself because the values are randomly assigned.

Algorithm 1 rGINs: GINs with random features

```
Input: $G = (V, E, \mathbf{X})$, Distribution $\mu$ of random feature, Parameters $\boldsymbol{\theta}$ of GINs.
Given embedding $[z_1, \ldots, z_n]^\top$, assign random feature $\mathbf{r}_v \sim \mu$ for all $v \in V$.
Return: $\mathrm{GIN}_{\boldsymbol{\theta}}\big((V, E, \mathrm{CONCAT}(\mathbf{X}, [\mathbf{r}_1, \ldots, \mathbf{r}_n]^\top))\big)$
```

4.2 Expressive power of GNNs with random feature

Based on the intuition above, Sato et al. (2020) proposes a new algorithm with random features named rGINs. Recall that Graph isomorphism networks(GINs) are a powerful model that takes a graph as input and outputs an embedding. rGINs assign random values to all nodes every time the procedure is called and calculate the node's embedding using GINs. They further demonstrate the expressive power of rGINs. Especially, Sato et al. (2020) proves that rGINs can distinguish any local structure with high probability. Another advantage of the random feature is that it can handle arbitrary large graphs. This is because rGINs draw random features from the same distribution, irrespective of the graph size.

5 Conclusion

In this report, we followed the line of Sato (2020) to figure out the relationship between the Graph Neural Networks and the local algorithms on distributed systems. It turns out that like the relationship between WL-test and GNN, GNN also has strong connection with local algorithms. In detail, Sato (2020) also proposed the XS-correspondence to better craft this connection. We refer readers to check papers in detail if interested.

References

Angluin, D. (1980). Local and global properties in networks of processors. In *Proceedings of the twelfth annual ACM symposium on Theory of computing.*

Corso, G., Cavalleri, L., Beaini, D., Liò, P. and Veličković, P. (2020). Principal neighbourhood aggregation for graph nets. *arXiv preprint arXiv:2004.05718* .

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B. and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. *arXiv preprint arXiv:1704.01665* .

Dasoulas, G., Santos, L. D., Scaman, K. and Virmaux, A. (2019). Coloring graph neural networks for node disambiguation. *arXiv preprint arXiv:1912.06058* .

Gori, M., Monfardini, G. and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2. IEEE.

Hella, L., Järvisalo, M., Kuusisto, A., Laurinharju, J., Lempiäinen, T., Luosto, K., Suomela, J. and Virtema, J. (2015). Weak models of distributed computing, with connections to modal logic. *Distributed Computing* **28** 31–53.

Linial, N. (1992). Locality in distributed graph algorithms. *SIAM Journal on computing* **21** 193–201.

Naor, M. and Stockmeyer, L. (1995). What can be computed locally? *SIAM Journal on Computing* **24** 1259–1277.

Sato, R. (2020). A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078* .

Sato, R., Yamada, M. and Kashima, H. (2019). Approximation ratios of graph neural networks for combinatorial problems. *arXiv preprint arXiv:1905.10261* .

Sato, R., Yamada, M. and Kashima, H. (2020). Random features strengthen graph neural networks. *arXiv preprint arXiv:2002.03155* .

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks* **20** 61–80.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I. and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer.

Suomela, J. (2013). Survey of local algorithms. *ACM Computing Surveys (CSUR)* **45** 1–40.

Xu, K., Hu, W., Leskovec, J. and Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* .

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L. and Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.*