Paper Sharing: How Powerful are K-hop Message Passing Graph Neural Networks

https://arxiv.org/abs/2205.13328

Yicheng Wang

Outline

1-Hop MPNN and 1-WL Test

Definition (1-Weisfeiler-Lehman Test)

1. Assign a same initial color $c^{(0)}$ to all nodes.
2. For $l = 0$ to L do:
   - $c^{(l+1)}(v) = \text{HASH}(c^{(l)}(v), \{c^{(l)}(u) | u \in N(v)\})$.

Facts from the GIN paper (Xu et al.):

- 1-hop MPNNs are at most as expressive as the 1-WL test.
- There exists some 1-hop MPNN (with injective message passing functions) that is as expressive as the 1-WL test.

How powerful is the 1-WL test?

Regular Graphs

Definition (Regular Graph)

A regular graph is a graph where each vertex has the same number of neighbors. A regular graph with vertices of degree r is called a r-regular graph.

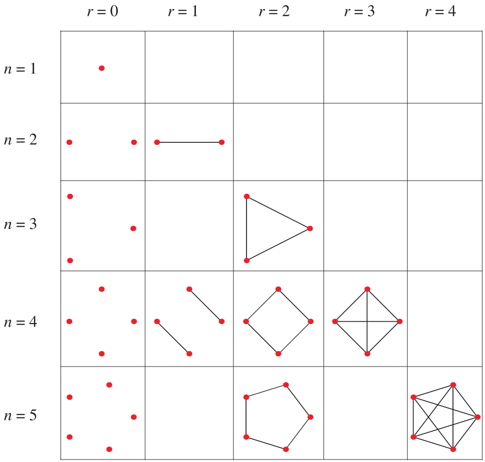

Figure: Credit: `https://mathworld.wolfram.com/RegularGraph.html`

How powerful is the 1-WL test?

Proposition 0

1-WL test cannot distinguish regular graphs with the same number of nodes and degree

Proof: For any d-regular graph with the same number of nodes:

- The initial colors are the same.
- If $c^{(l)}(v)$ are the same for all nodes, then the sets of neighbors' colors are the same for all nodes. Thus, $c^{(l+1)}(v)$ are the same.
- Thus, the 1-WL kernels are the same for all such graphs.

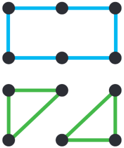

Figure 1: Two graphs not distinguished by 1-WL.

Figure: Credit: Provably Powerful Graph Networks, Maron et al.

K-Hop Neighborhood

What if we also collect k-th hop neighborhood for $k = 1, 2, \ldots, K$?

K-Hop Neighborhood

What if we also collect k-th hop neighborhood for $k = 1, 2, \ldots, K$?
But there are different definitions of the k-th hop neighbors. The paper differentiated two kernels:

1. shortest path distance (spd) kernel
2. and graph diffusion (gd) kernel.

In the followings, we deonte by $Q^{k,t}_{v,G}$ the k-th hop neighbors (exact) of node v in graph G in the sense of kernel t.

Two Kernels

- $Q_{v,G}^{k,spd}$ = the set of neighbors that are at a distance of k from v.
- $Q_{v,G}^{k,gd}$ = the set of neighbors that can diffuse information to node v with k diffusion steps.

Graph Diffusion in GNNs (from DGL Docs)

Mathematically, let $\vec{x}$ be the vector of node signals, then a graph diffusion operation can be defined as $\vec{y} = \tilde{A}\vec{x}$, where $\tilde{A}$ is the diffusion matrix.

The demonstrations in the paper use the adjacency matrix as the diffusion matrix, i.e., $Q_{v,G}^{k,gd}$ is the set of neighbors reachable from v with k steps.

Two Kernels: Example

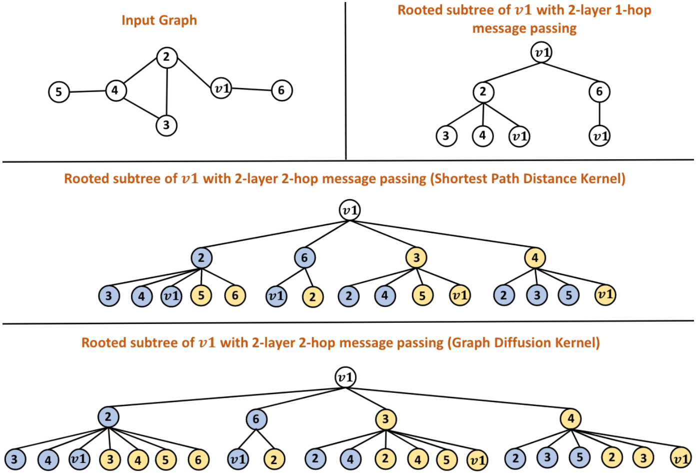

Figure: Illustration

Blue nodes are 1st hop neighbors of the parent node.
Yellow nodes are the 2nd hop neighbors of the parent node.

K-Hop Message Passing Framework

K-hop message passing can be formulated as:

$$\vec{m}_v^{l,k} = \mathrm{MES}_k^l(\{(\vec{h}_u^{l-1}, e_{uv}) | u \in Q_{v,G}^{k,t}\}),$$
$$\vec{h}_v^{l,k} = \mathrm{UPD}_k^l(\vec{m}_v^k, \vec{h}_v),$$
$$\vec{h}_v^l = \mathrm{COMBINE}^l(\{\vec{h}_v^{l,k} | k = 1, 2, \ldots, K\}),$$

where l is the layer number and k represents the hop number.
The COMBINE function combines the representations of node v at different hops.

K-Hop Message Passing Framework

K-hop message passing can be formulated as:

$$\vec{m}_v^{l,k} = \text{MES}_k^l(\{(\vec{h}_u^{l-1}, e_{uv}) | u \in Q_{v,G}^{k,t}\}),$$
$$\vec{h}_v^{l,k} = \text{UPD}_k^l(\vec{m}_v^k, \vec{h}_v),$$
$$\vec{h}_v^l = \text{COMBINE}^l(\{\vec{h}_v^{l,k} | k = 1, 2, \ldots, K\}),$$

where l is the layer number and k represents the hop number.
The COMBINE function combines the representations of node v at different hops.

Proposition 3

The map $\{(k, \{(\vec{h}_u^{l-1}, e_{uv}) | u \in Q_{v,G}^{k,t}\}, \vec{h}_v^{l-1}) | k = 1, 2, \ldots, K\} \mapsto \vec{h}_v^l$ is injective.

Proof:

- We know the existence of injective message functions (MES) and injective updating functions (UPD).
- The hop number can be encoded into MES_k / UPD_k their ouputs (e.g., simply append the number k to the output vector).
- Thus, the overall mapping is injective.

K-Hop Message Passing Framework

K-hop message passing can be formulated as:

$$\vec{m}_v^{l,k} = \text{MES}_k^l(\{(\vec{h}_u^{l-1}, e_{uv}) | u \in Q_{v,G}^{k,t}\}),$$
$$\vec{h}_v^{l,k} = \text{UPD}_k^l(\vec{m}_v^k, \vec{h}_v),$$
$$\vec{h}_v^l = \text{COMBINE}^l(\{\vec{h}_v^{l,k} | k = 1, 2, \ldots, K\}),$$

where l is the layer number and k represents the hop number.
The COMBINE function combines the representations of node v at different hops.

Proposition 3

The map $\{(k, \{(\vec{h}_u^{l-1}, e_{uv}) | u \in Q_{v,G}^{k,t}\}, \vec{h}_v^{l-1}) | k = 1, 2, \ldots, K\} \mapsto \vec{h}_v^l$ is injective.

Proof:

- We know the existence of injective message functions (MES) and injective updating functions (UPD).
- The hop number can be encoded into MES_k / UPD_k their ouputs (e.g., simply append the number k to the output vector).
- Thus, the overall mapping is injective.

We say a K-hop MPNN is "proper" if all the message, update, and combine functions are all injective given the input from a countable space.

K-Hop MPNN is Strictly More Powerfull Than 1-WL

Proposition 1

A proper K-hop MPNN ($K > 1$) is strictly more powerful than 1-WL test.

Proof:

1. The proper K-hop message passing at each layer provides richer information than 1-WL.
2. There are examples in which the proper K-hop MPNN is better.

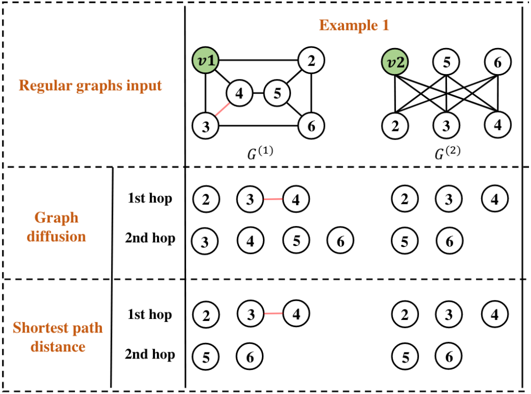

Figure: Example

A Stronger Statement

Theorem (1)

Let n, r be integers (r satisfies some condition). With at most $K = K_0$, there exists a 1-layer K-hop MPNN with the spd kernel that can distinguish almost all pairs of n-sized r-regular graphs.

A Stronger Statement

Theorem (1)

Let n, r be integers (r satisfies some condition). With at most $K = K_0$, there exists a 1-layer K-hop MPNN with the spd kernel that can distinguish almost all pairs of n-sized r-regular graphs.

- $3 \leq r \leq (2 \log 2n)^{1/2}$.
- $K_0 = \lfloor \left(\frac{1}{2} + \epsilon\right) \frac{\log 2n}{\log(r-1)} \rfloor$.
- "almost all" means $1 - o(n^{-1/2})$ probability.
- ϵ is a constant used in another paper the authors cited to prove this theorem.

See the paper for theoretical analysis on existing GNNs using the K-hop message passing framework.

Upperbound

Theorem (2)

A proper K-hop MPNN with any kernel is at most as expressive as the 3-WL test.

Upperbound

Theorem (2)

A proper K-hop MPNN with any kernel is at most as expressive as the 3-WL test.

Understanding and Extending Subgraph GNNs by Rethinking Their Symmetries (Frasca et al.) showed that:

Lemma 1

Any subgraph-based GNNs with node-based selection policy can be implemented by 3-IGN and thus their expressive power is bounded by 3-WL test.

Proof: Can show that K-hop MPNNs with any of the two kernels can also be implemented by 3-IGN. (The authors directly follow all the definitions and notations in Frasca et al.'s paper).
In summary, the K-hop MPNNs' expressivity: between 1-WL (2-WL) and 3-WL.

How Powerful is 3-WL Test?

Facts:

- 1-WL and 2-WL have equivalent expressivity. (Provably Powerful Graph Networks, Maron et al.)
  The Pure Transformer paper (Kim et al.) used the term 2-WL.
- 3-WL test cannot distinguish non-isomorphic distance-regular graphs.

How Powerful is 3-WL Test?

Facts:

- 1-WL and 2-WL have equivalent expressivity. (Provably Powerful Graph Networks, Maron et al.)
  The Pure Transformer paper (Kim et al.) used the term 2-WL.
- 3-WL test cannot distinguish non-isomorphic distance-regular graphs.

Definition (Distance-Regular Graph)

A regular graph such that for any u, v the number of vertices "at distance j from u and at distance k from v" depends only on j, k, and the distance between u and v.

Definition (Intersection Array)

A graph of diameter d is distance-regular iff there exists (characterized by) an intersection array $(b_0, b_1, \ldots, b_{d-1}; c_1, \ldots, c_d)$ such that for $j = 1, \ldots, d$, for any pair of u, v with distance j:

- b_j = # neighbors of u at distance $j+1$ from v,
- c_j = # neighbors of u at distance $j-1$ from v.

Distance-Regular Graph Example

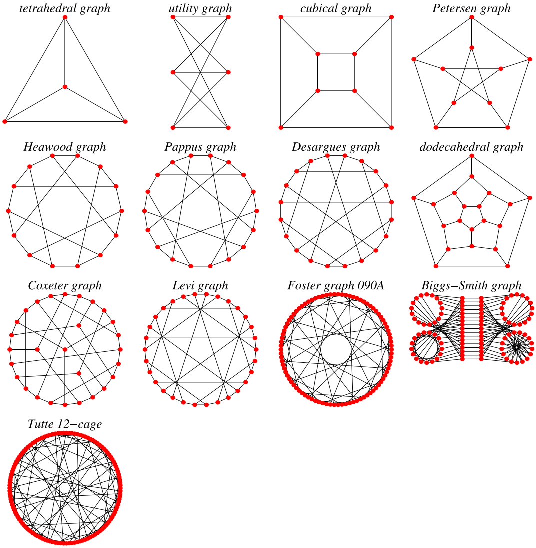

Figure: Credit: `https://mathworld.wolfram.com/Distance-RegularGraph.html`

Peripheral Subgraph

Motivation:

- K-hop MPNNs capture the heirarchy of neighbors at each hop.
- What if we also capture the connectivity among neighbors at each hop?

Peripheral Subgraph

Motivation:

- K-hop MPNNs capture the heirarchy of neighbors at each hop.
- What if we also capture the connectivity among neighbors at each hop?

Definition (Peripheral Edge and Subgraph)

The peripheral edge $E(Q^{k,t}_{v,G})$ is defined as the set of edges among nodes in $Q^{k,t}_{v,G}$.
The peripheral subgraph $G^{k,t}_{v,G}$ is defined as the subgraph in G induced by $Q^{k,t}_{v,G}$.

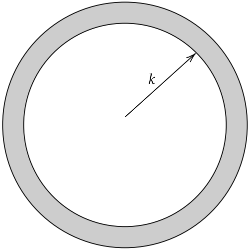

Figure: Illustration

Peripheral Subgraph Example

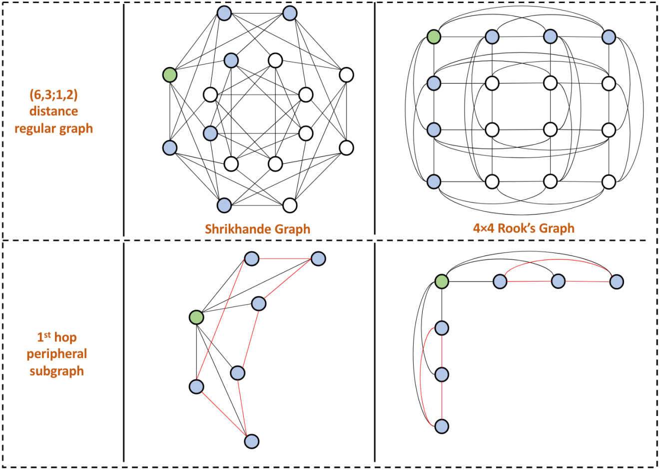

Figure: Example

The subgraphs are exactly the same as How powerful is the 1-WL test? . Another Example showing its superior expessivity than neighbor sets.

KP-GNN Framework

- The paper introduce the K-hop Peripheral-subgraph-enhanced GNN (KP-GNN), designed to improve the expressivity of K-hop MPNN.
- The only difference: From

$$\vec{m}_v^{l,k} = \mathrm{MES}_k^l(\{(\vec{h}_u^{l-1}, e_{uv}) | u \in Q_{v,G}^{k,t}\})$$

to

$$\vec{m}_v^{l,k} = {}^{\mathrm{KP}}\mathrm{MES}_k^l(\{(\vec{h}_u^{l-1}, e_{uv}) | u \in Q_{v,G}^{k,t}\},\ G_{v,G}^{k,t}).$$

- This can be regarded as a method of feature augmentation.

How to Encode $G_{v,G}^{k,t}$

Encode some counting information about $G_{v,G}^{k,t}$:

- neighbor counts: counts of i-th hop neighbors of each node in $G_{v,G}^{k,t}$, where $i = 1, \ldots, k'$.
- edge counts: counts of peripheral edges of each node in $G_{v,G}^{k,t}$ at the i-th hop (up to k'), $E(Q^{i,t}_{u,G^{k,t}_{v,G}})$.
- Sums them up (along each hop?) to constrain the dimension.
- Can be preprocessed and reused.

How to Encode $G_{v,G}^{k,t}$

Encode some counting information about $G_{v,G}^{k,t}$:

- neighbor counts: counts of i-th hop neighbors of each node in $G_{v,G}^{k,t}$, where $i = 1, \ldots, k'$.
- edge counts: counts of peripheral edges of each node in $G_{v,G}^{k,t}$ at the i-th hop (up to k'), $E(Q^{i,t}_{u,G_{v,G}^{k,t}})$.
- Sums them up (along each hop?) to constrain the dimension.
- Can be preprocessed and reused.

Equivalent to running another 1-layer KP-GNN for $G_{v,G}^{k,t}$? Peripheral Subgraph Example
$\implies$ higher chance to distinguish neighbor structure $\implies$ higher change to distinguish the whole graph.

Details

The counting information is called k'-configuration and is denoted by $C_k^{k'}$.
Implementation:

$$f(G_{v,G}^{k,t}) = \text{EMB}(E(Q_{v,G}^{k,t}), C_k^{k'}),$$
$$^{\text{KP}}\text{MES}_k^l(\square, G_{v,G}^{k,t}) = \text{MES}_k^l(\square) + f(G_{v,G}^{k,t}),$$

where

- EMB is an learnable embedding function.

Generalizable to other graphs.

Expressivity of KP-GNN

Proposition 2

A proper 1-layer d-hop KP-GNN (with the specific implementation) can distinguish two non-isomorphic distance-regular graphs with the same intersection array if the k'-configurations of the two graphs at some hop (1 to the diameter) are different.

- A relatively weak proposition.

Peripheral Subgraph Example

In summary, KP-GNNs' expressivity:

- > K-hop MPNN (because it encodes additional information),
- sometimes > 3-WL (at distinguishing distance-regular graph),
- upper bound unknown.

Limitation

An intrinsic limitation that exists in K-hop/KP- GNNs:

- Using K-hop instead of 1-hop can make the receptive field of a node increase with K.
- According to the GINE+ paper (Brossard et al.): The increased receptive field can hurt learning.

Space and Time Complexity

The K-hop message passing and KP-GNN:

- Both have the space complexity of $O(n)$: Do not add additional space complexity because of more hops.
  - in terms of # (final?) representations for all nodes.
- Both have time complexity $O(n^2)$ for the shortest path distance kernel, which is faster than subgraph-based GNNs (at least $O(nm)$) but slower than MPNN ($O(m)$).
  - in terms of # node-edge pairs involved in one run of message passing (each node involves at most n nodes)
  - The counting information is preprocessed and will be amortized to zero finally.

Remark:

1. Have to store $\vec{m}_v^{l,k}$ for each hop k during the computation.

Experiment

Table 1: Empirical evaluation of the expressive power.

Method	K	EXP (ACC) SPD	EXP (ACC) GD	SR (ACC) SPD	SR (ACC) GD	CSL (ACC) SPD	CSL (ACC) GD
K-GIN	K=1	50	50	6.67	6.67	12	12
	K=2	50	50	6.67	6.67	32	22.7
	K=3	100	66.9	6.67	6.67	62	42
	K=4	100	100	6.67	6.67	92.7	62.7
KP-GIN	K=1	50	50	100	100	22	22
	K=2	100	100	100	100	52.7	52.7
	K=3	100	100	100	100	90	90
	K=4	100	100	100	100	100	100

Figure: Expressivity

Table 3: TU dataset evaluation result.

Method	MUTAG	D&D	PTC-MR	PROTEINS	IMDB-B
WL	90.4±5.7	79.4±0.3	59.9±4.3	75.0±3.1	73.8±3.9
GIN	89.4±5.6	-	64.6±7.0	75.9±2.8	75.1±5.1
DGCNN	85.8±1.7	79.3 ±0.9	58.6 ±2.5	75.5±0.9	70.0±0.9
GraphSNN	91.24±2.5	82.46±2.7	66.96±3.5	76.51±2.5	76.93±3.3
GIN-AK+	91.30±7.0	-	68.20±5.6	77.10±5.7	75.60±3.7
KP-GCN	91.7±6.0	79.0±4.7	67.1±6.3	75.8±3.5	75.9±3.8
KP-GraphSAGE	91.7±6.5	78.1±2.6	66.5±4.0	76.5±4.6	76.4±2.7
KP-GIN	92.2±6.5	79.4±3.8	66.8±6.8	75.8±4.6	76.6±4.2
GIN-AK+*	95.0±6.1	OOM	74.1±5.9	78.9±5.4	77.3±3.1
GraphSNN*	94.70±1.9	**83.93±2.3**	70.58±3.1	78.42±2.7	78.51±2.8
KP-GCN*	**96.1±4.6**	83.2±2.2	**77.1±4.1**	80.3±4.2	79.6±2.5
KP-GraphSAGE*	**96.1±4.6**	83.6±2.4	76.2±4.5	**80.4±4.3**	80.3±2.4
KP-GIN*	95.6±4.4	83.5±2.2	76.2±4.5	79.5±4.4	**80.7±2.6**

Figure: Real-world dataset (e.g., TU)

Table 2: Simulation dataset result. The top two are highlighted by **First**, **Second**.

Method	Node Properties ($\log_{10}$(MSE))			Graph Properties ($\log_{10}$(MSE))			Counting Substructures (MAE)			
	SSSP	Ecc.	Lap.	Connect.	Diameter	Radius	Tri.	Tailed Tri.	Star	4-Cycle
GIN	-2.0000	-1.9000	-1.6000	-1.9239	-3.3079	-4.7584	0.3569	0.2373	0.0224	0.2185
PNA	**-2.8900**	**-2.8900**	-3.7700	-1.9395	3.4382	-4.9470	0.3532	0.2648	0.1278	0.2430
PPGN	-	-	-	-1.9804	-3.6147	-5.0878	**0.0089**	**0.0096**	**0.0148**	**0.0090**
GIN-AK+	-	-	-	**-2.7513**	**-3.9687**	-5.1846	0.0123	0.0112	**0.0150**	**0.0126**
K-GIN+	-2.7919	-2.5938	**-4.6360**	-2.1782	**-3.9695**	**-5.3088**	0.2593	0.1930	0.0165	0.2079
KP-GIN+	**-2.7969**	**-2.6169**	**-4.7687**	**-4.4322**	-3.9361	**-5.3345**	**0.0060**	**0.0073**	0.0151	0.0395

Figure: Counting Graph Properties and Substructures