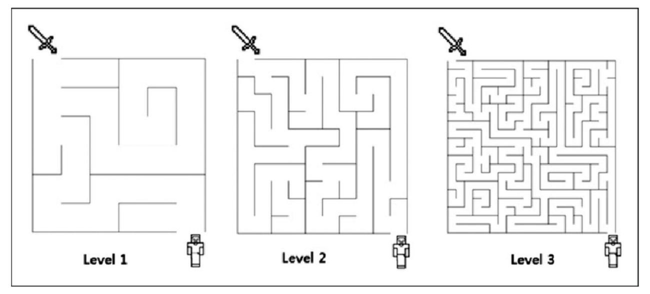
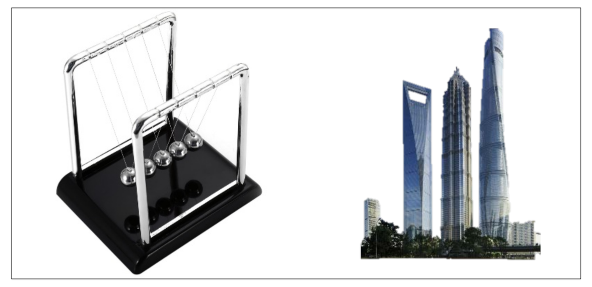
Towards Scale-Invariant Graph-related Problem Solving by Iterative Homogeneous Graph Neural Networks

Hao Tang, Zhiao Huang, Jiayuan Gu, Bao-Liang Lu, Hao Su

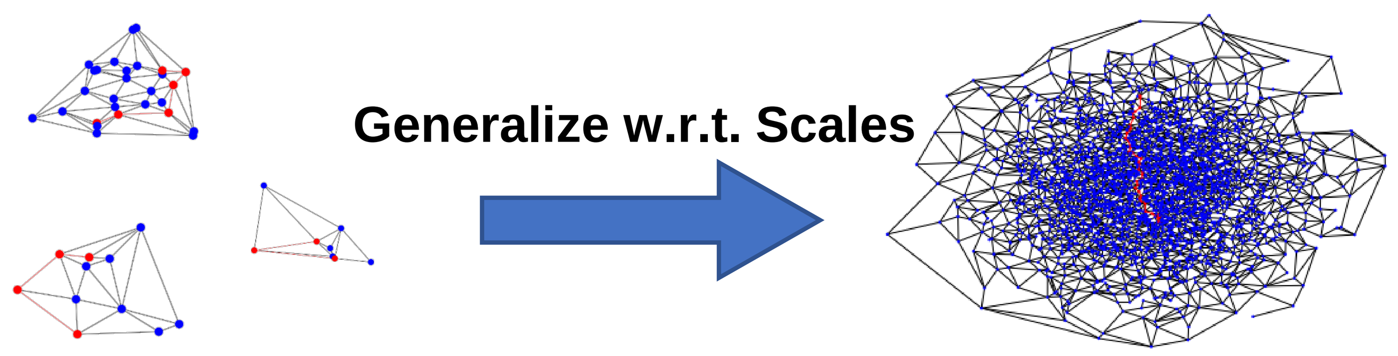
Problem Setup: generalize w.r.t. graph scales

Problems met in practice are of diversified scales. People can learn knowledge from small-scale data and apply it to much larger-scale data.





Given the expressiveness of graphs, many problems can be reduced to graph-related problems. Therefore, we want models that can **generalize w.r.t. graph scales (graph sizes, graph diameters, edge weights, etc.).**



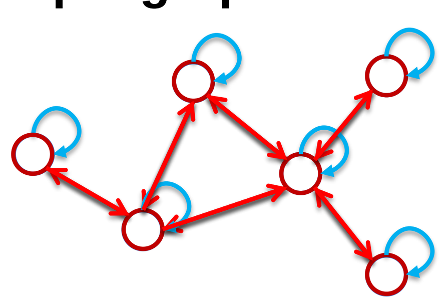
Trained on Smaller Graphs (e.g. ≤ 40 nodes)

Tested on Larger Graphs (e.g. 5000 nodes)

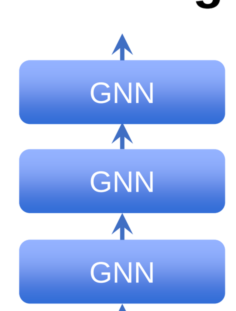
Current GNNs lack such generalizability

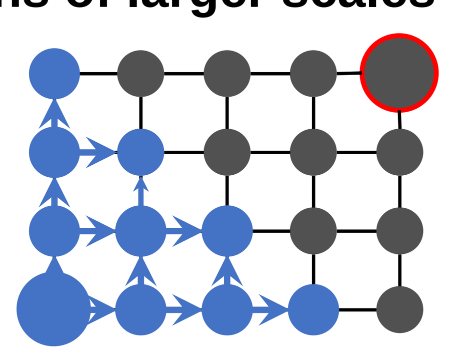
Current GNNs lack the generalizability w.r.t. graph scales (graph sizes, graph diameters, edge weights, etc.) for at least two reasons:

- **GNNs with restricted depth and width cannot solve many simple graph-related problems on graphs of larger scales** [1].



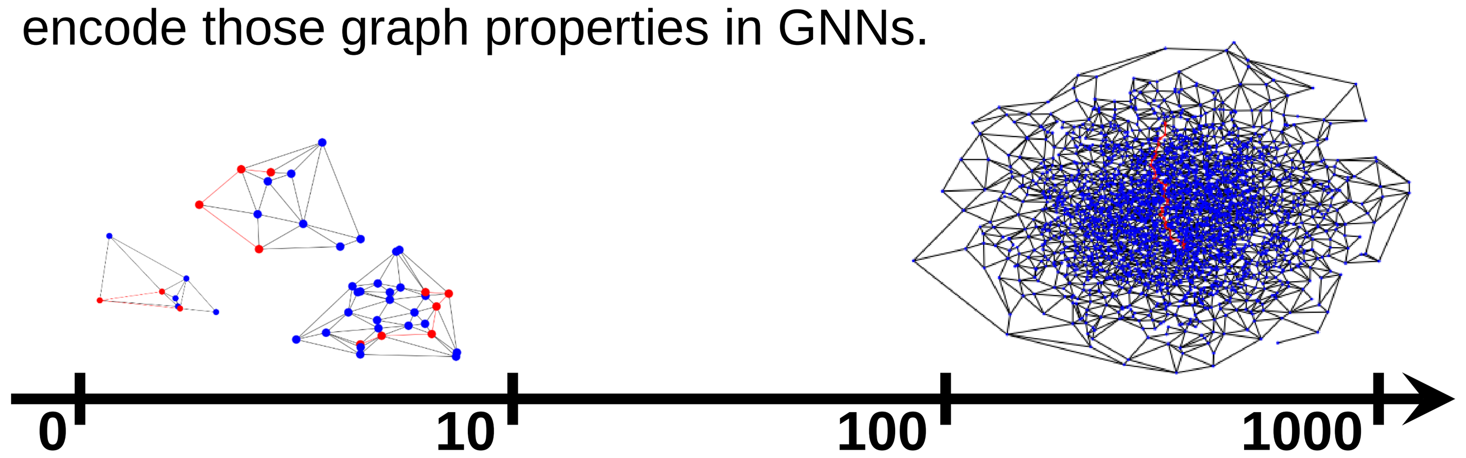
The locality nature of GNNs (i.e., message passing)





Shallow GNNs **cannot** even send messages between nodes whose distance are large.

- **The graph properties deviate greatly for graphs of different scales**, thus the distributions of the internal representations that encode those graph properties in GNNs.



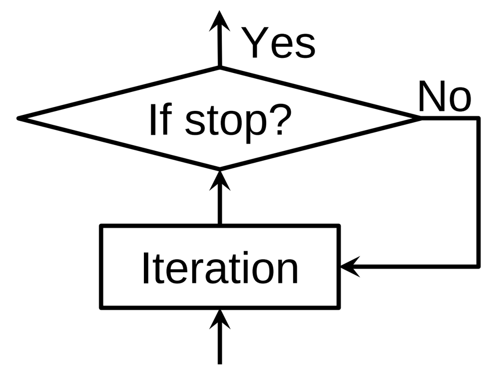
Graph properties to be encoded by internal representations (e.g., distances)

And **the performance of classical neural network modules** (e.g., the MLPs in GNNs) **are usually highly degraded on those out-of-range inputs.**

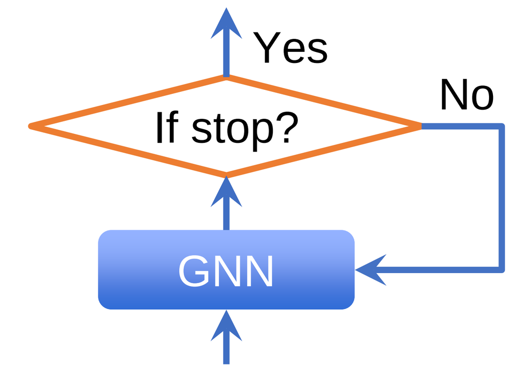
Iterative Graph Neural Networks

To address the fixed layer number issue, we introduce IterGNN to achieve adaptive and unbounded computation steps.

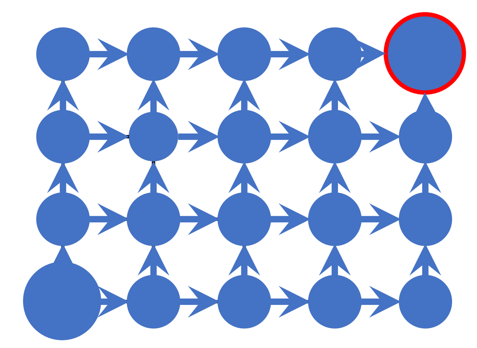
- **Typical graph algorithms,** e.g., Dijkstra's algorithm for shortest path computation, **are iterative so that they can handle problems of arbitrary scales.**



Iterative Algorithm

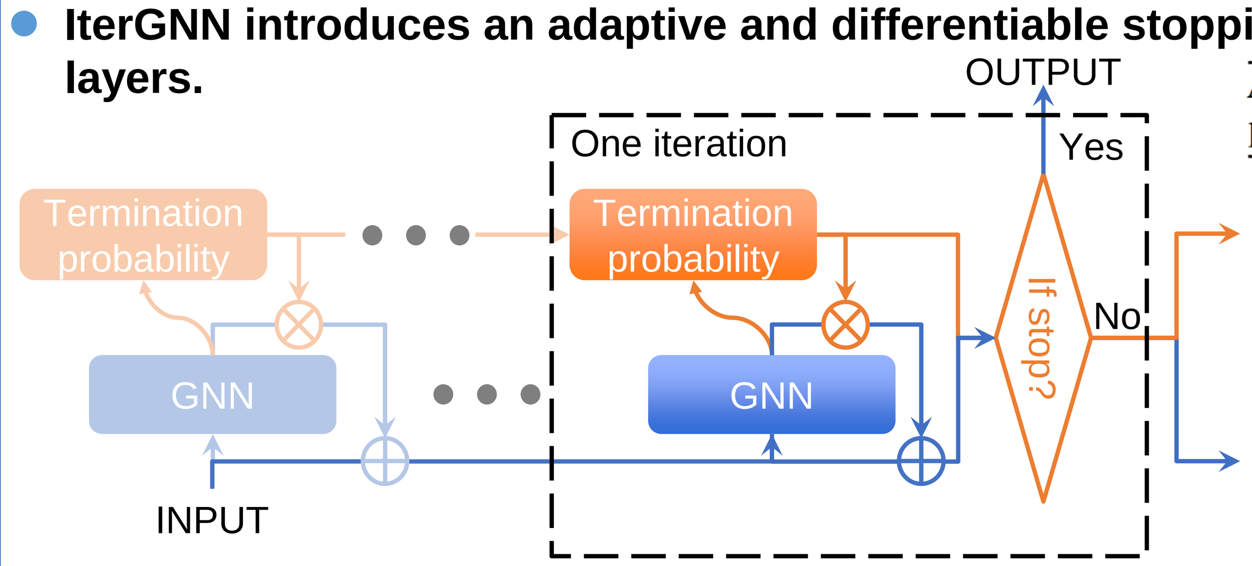


general illustration of Iterative GNN



IterGNN **can** generalize to graphs of larger scales

- **IterGNN introduces an adaptive and differentiable stopping criterion to allow flexible GNN layers.**



Algorithm 1: Iterative module. g is the stopping criterion and f is the iteration body

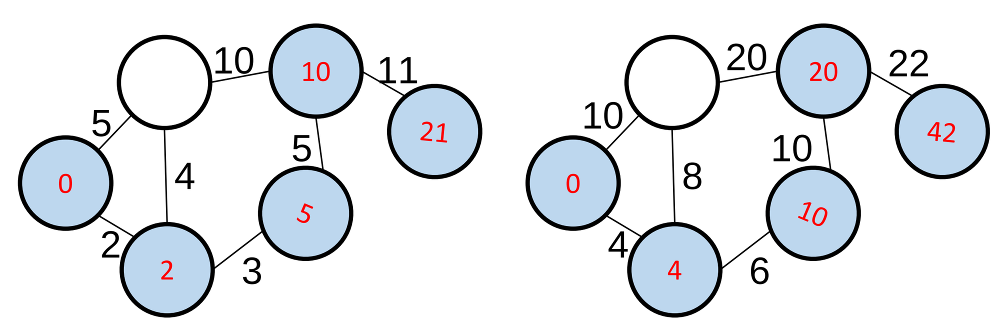
input: initial feature x; stopping threshold ϵ
$k \leftarrow 1$
$h^0 \leftarrow x$
while $\prod_{i=1}^{k-1}(1-c^i) > \epsilon$ **do**
  $h^k \leftarrow f(h^{k-1})$
  $c^k \leftarrow g(h^k)$
  $k \leftarrow k+1$
end while
return $h = \sum_{j=1}^{k-1}\left(\prod_{i=1}^{j-1}(1-c^i)\right)c^j h^j$

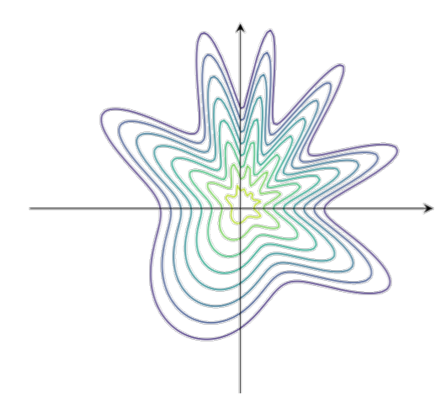
- **Our stopping condition is** adaptive to the inputs, supports arbitrarily large iteration numbers, and, interestingly, is able to *be trained in an end-to-end fashion without any direct supervision*.

Homogeneous Graph Neural Networks

To address the issue of out-of-range encoding, we notice that

- **The solutions to many graph-related problems,** such as shortest path, TSP, and maximum flow, **are homogeneous w.r.t. the input graph weights,** i.e., the solution scales linearly with the magnitudes of the input weights.



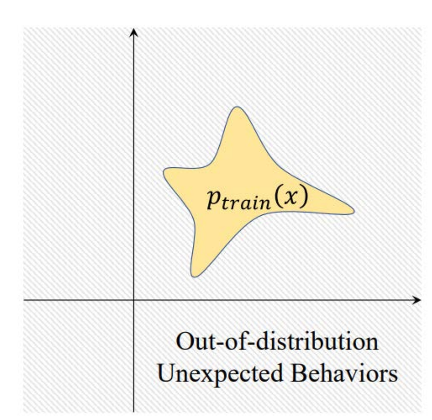


Solutions to shortest path are homogeneous.
(i.e., if we multiply the edge weights by 2, the shortest path lengths are also multiplied by 2.)

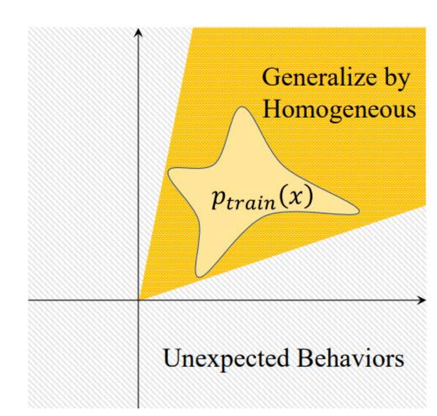
$f(\lambda x) \equiv \lambda f(x), \forall \lambda > 0, x \in R^2$
(another example of homogeneous functions)

We prove that

- Generalization errors are bounded if both neural networks and the target function are homogeneous under proper conditions.
- Universal approximators of homogeneous functions are easy to build, e.g., $Relu(Wx+b) \Rightarrow Relu(Wx)$ in MLPs.



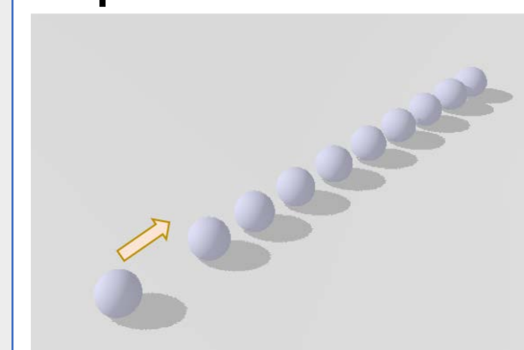
Behaviors of MLP



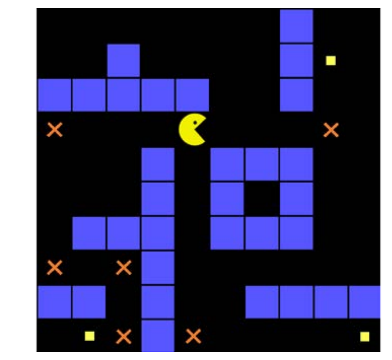
Behaviors of HomoMLP

Experiment Results

Graph-related Problems & Tasks:



Physical Simulation



Symbolic Pacman

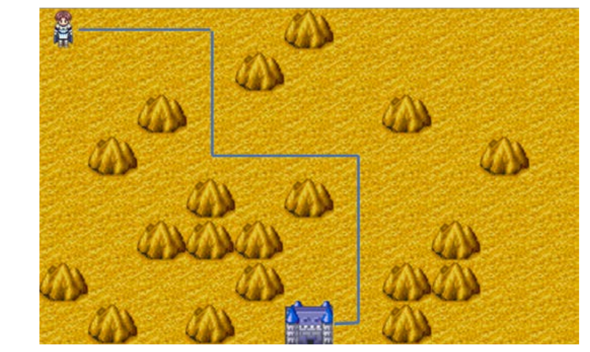


Image-based Navigation

Compared Methods:

- *GCN, GAT, Path:* stack 30 GCN/GAT/Path-GNN layers to build the model.
- *Homo-Path:* apply the homogeneous prior to the "Path" model.
- *Iter-Path:* adopt the iterative module to control the iteration number of the GNN layer in the "Path" model.
- *Iter-Homo-Path:* integrate all proposals.

Our model, Iter-Homo-Path, successfully generalize w.r.t. graph scales.

Table 1: Generalization performance on graph algorithm learning and graph-related reasoning. Models are trained on graphs of smaller sizes (e.g., within $[4, 34)$ or $\leq 10 \times 10$) and are tested on graphs of larger sizes (e.g., 50, 100, 500, 16×16 or 33×33). The metric for the shortest path and TSP is the relative loss. The metric for component counting is accuracy. The metric for physical simulation is the mean square error. The metric for image-based navigation is the success rate.

	Graph Theory Problems					Graph-related Reasoning			
	Shortest Path		Component Cnt.		TSP	Physical sim.		Image-based Navi.	
Models	ER	Lob	ER	Lob	2D	50	100	16×16	33×33
GCN [38]	0.1937	0.44	0.0%	0.0%	0.52	42.18	121.14	34.2%	28.9%
GAT [39]	0.1731	0.28	24.4 %	0.0%	0.18	>1e4	>1e4	56.7%	44.5%
Path (ours)	**0.0003**	0.29	82.3%	77.2%	0.16	20.24	27.67	85.6%	65.1%
Homo-Path (ours)	0.0008	0.27	91.9%	83.9%	0.14	20.48	21.45	87.8%	69.3%
Iter-Path (ours)	0.0005	0.09	86.7%	96.1%	0.08	**0.13**	**1.68**	89.4%	78.6%
Iter-Homo-Path (ours)	0.0007	**0.02**	**99.6%**	**97.5%**	**0.07**	**0.07**	**2.01**	**98.8%**	**91.7%**

Table 2: Generalization performance on the shortest path problem with lobster graphs. During training, node numbers are within $[4, 34)$ for unweighted problems (whose metric is the success rate), and edge weights are within $[0.5, 1.5)$ for weighted problems (whose metric is the relative loss).

Generalize	w.r.t. sizes and diameters - unweighted					w.r.t. magnitudes - weighted			
	20	100	500	1000	5000	$[0.5, 1.5)$	$[1, 3)$	$[2, 6)$	$[8, 24)$
GCN [38]	66.6	25.7	5.5	2.4	0.4	0.31	0.37	0.49	0.56
GAT [39]	**100.0**	42.7	10.5	5.3	0.9	0.13	0.29	0.49	0.55
Path (ours)	**100.0**	62.9	20.1	10.3	1.6	0.06	0.22	0.44	0.54
Homo-Path (ours)	**100.0**	58.3	53.7	50.2	1.6	0.03	**0.03**	**0.03**	**0.03**
Iter-Homo-Path (ours)	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**0.01**	0.04	0.06	0.08

Ablation Studies show that each of our components is beneficial.

Iter-Homo-Path	
100.0	
Homo-Path	Iter-Path
53.7	48.9
ACT-Homo-Path	Iter-Homo-GAT
52.7	2.9
Shared-Homo-Path	Iter-Homo-GCN
91.7	1.4

Table 3: Ablation studies of generalization performance for the shortest path problem on lobster graphs with 1000 nodes. Metric is the success rate.

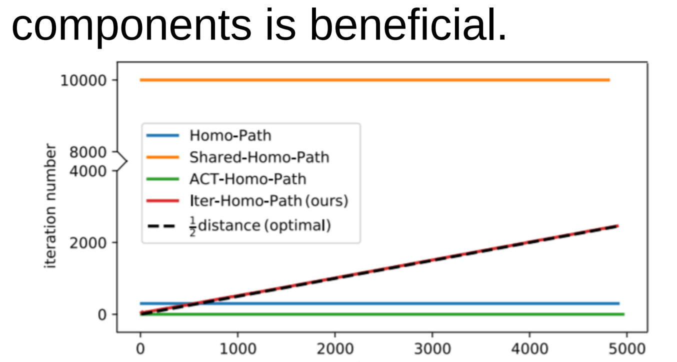


Figure 4: The iteration numbers of GNN layers w.r.t. the distances from the source node to the target node for the shortest path problem.

Interpretable Behaviors: Our model learned an optimal stopping criterion to schedule the iterations for the unweighted shortest path problem.

Reference

Tang, H., Z. Huang, J. Gu, B.-L. Lu, H. Su. Towards Scale-Invariant Graph-related Problem Solving by Iterative Homogeneous Graph Neural Networks, NeurIPS 2020.

[1] Loukas, A. What graph neural networks cannot learn: depth vs width. In International Conference on Learning Representations. 2020.