

# K-hop Graph Neural Networks

Presented by  
Lee Guan Bo Ambrose



UNIVERSITY OF  
**WATERLOO**

DAVID R. CHERITON SCHOOL  
OF COMPUTER SCIENCE



SCHOOL OF  
**COMPUTING &  
DATA SCIENCE**  
The University of Hong Kong

# **k-hop graph neural networks**

Giannis Nikolentzos<sup>1,2</sup>, George Dasoulas<sup>1,3</sup> and Michalis Vazirgiannis<sup>1,2</sup>

<sup>1</sup>École Polytechnique, France

<sup>2</sup>Athens University of Economics and Business, Greece

<sup>3</sup>Noah's Ark Lab, Huawei, France

<https://arxiv.org/pdf/1907.06051>

(Nikolentzos et al., 2020)

# Outline

- Introduction
- Preliminaries
- Limitation of standard GNNs
- Better architecture: k-hop GNNs
- Evaluation on datasets
- Conclusion

# Part 1: Introduction

# Background

- Graph Neural Networks (GNNs) have been used in many tasks
  - Exp: Chemoinformatics, social networks, traffic systems...
  - Node classification, link prediction, graph-related tasks
- Achieved SOTA performance in different benchmarks

# What is the problem?

1. Uncertain whether GNNs can identify fundamental graph properties
  - Expressive power of GNNs are similar to WL test (in terms of distinguishing non-isomorphic graphs)
  - WL subtree kernel cannot effectively identify fundamental graph properties (Kriege et al., 2018)
2. If not, we need architecture that can give more powerful representation



# Why is it important?

- Quality of the feature vector  $\propto$  Representation power and expressiveness
- Performance of each task  $\propto$  Quality of the feature vector

# A glimpse on the paper

- Analyze the representational power of GNNs
  - Failed to identify fundamental properties
- Propose new architecture (k-hop GNNs)
  - Proved to be more powerful than standard GNNs
- Evaluate its performance on different datasets
  - Better than most SOTA models



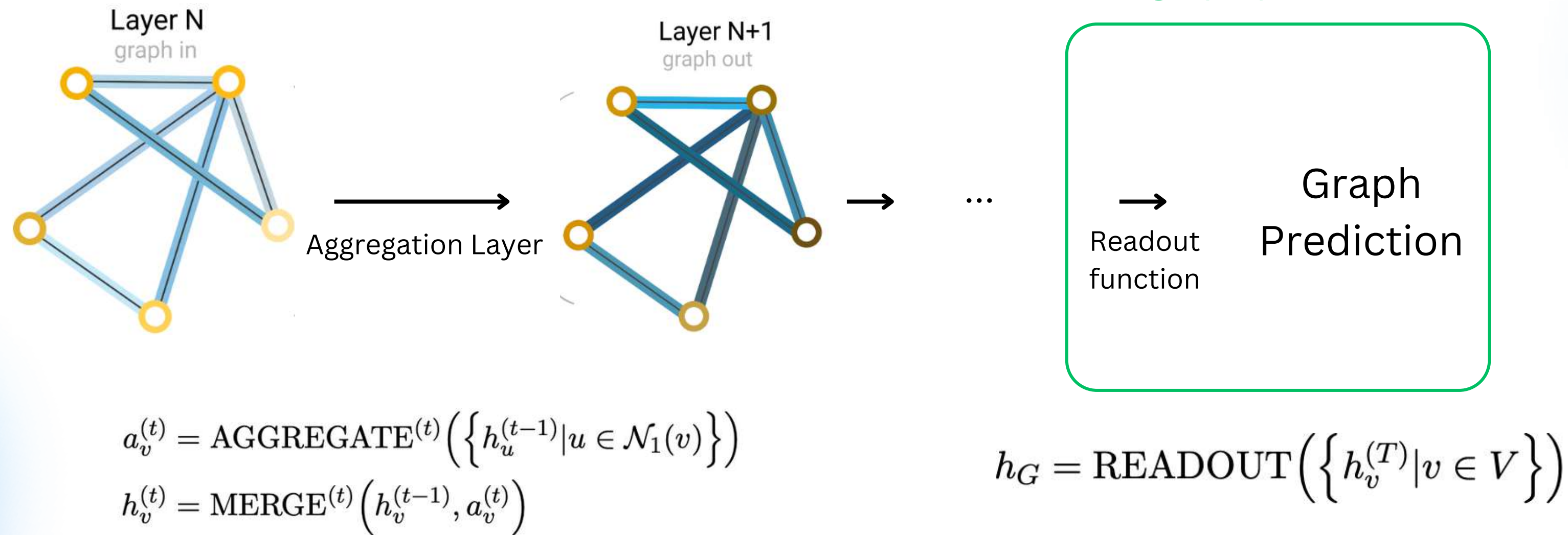
# Part 2: Preliminaries

# Notations

Notation	Explanation
$G = (V, E)$	G: undirected graph, V: set of nodes; E set of edges
$n, m$	n: # of nodes; m: # of edges
$N_k(v)$	The set of neighbours of node v within radius k
$G_v^k$	Subgraph induced by node v and its k-hop neighbour
$\mathcal{L}$	A set of labels
$h_v^{(t)}$	Feature vector of node v at layer t

# Standard GNNs

(Sanchez-Lengeling et al., 2021)



Example of AGGREGATE/READOUT: SUM, AVG, MAX

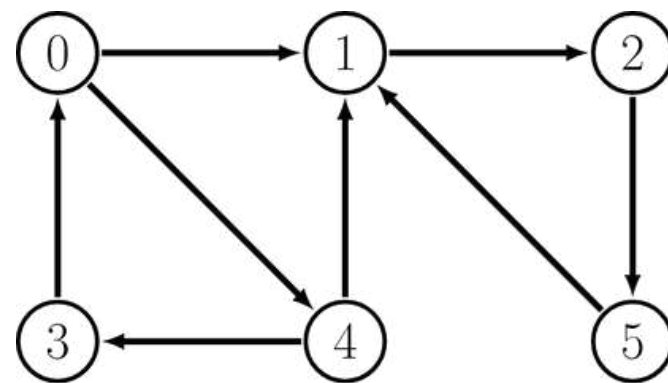
Example of MERGE: MLP()

# Identification & 3 graph features

## Identification

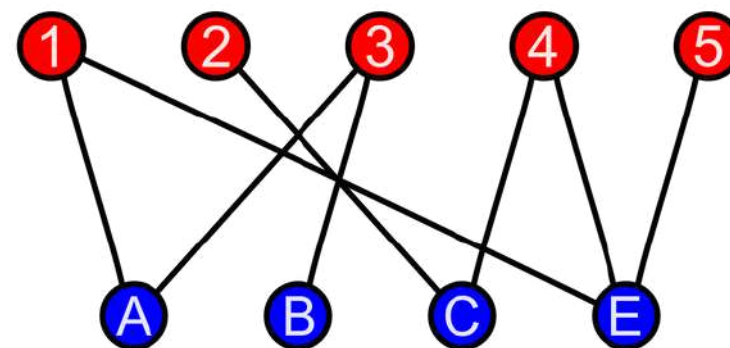
Same representation only if 2 graphs both have or do not have that property

## 3 fundamental properties



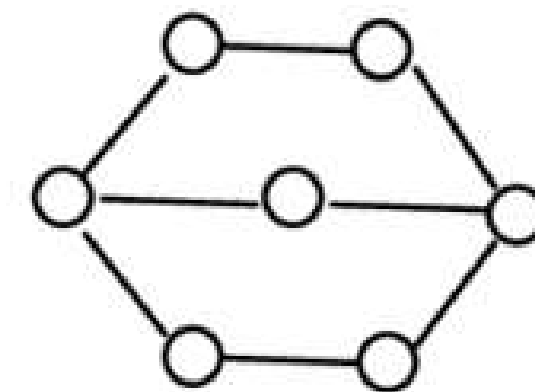
### **Connectivity**

Any vertex can reach any other vertex



### **Bipartiteness**

Can decompose  $V$  into disjoint  $V_1$  and  $V_2$ , such that every edge connects a vertex in  $V_1$  to a vertex in  $V_2$



### **Triangle-freeness**

Does not contain any triangle

# Part 3: Limitation of Standard GNNs



# Limitation of standard GNNs (1)

- Assume all nodes with same degree have the same feature vector
  - In regular graphs, all nodes have the same feature vector
  - Regular graph: All nodes have the same degree

Standard GNNs produce the same representation for the nodes of all regular graphs of a specific size and degree  
(proved in next page)



# Limitation of standard GNNs (2)

Setting: Two non-isomorphic regular graphs  $G_1 = (V_1, E_1)$   
 $G_2 = (V_2, E_2)$   
We want to prove  $h_{v_1}^{(t)} = h_{v_2}^{(t)}$

- Using Mathematical Induction
  - At  $t=0$ ,
    - $h_{v_1}^{(0)} = h_{v_2}^{(0)}$  (all nodes have same initial representations)

# Limitation of standard GNNs (2)

Setting: Two non-isomorphic regular graphs  $G_1 = (V_1, E_1)$   
 $G_2 = (V_2, E_2)$   
We want to prove  $h_{v_1}^{(t)} = h_{v_2}^{(t)}$

- Using Mathematical Induction

- Assume  $h_{v_1}^{(t-1)} = h_{v_2}^{(t-1)}$  (for induction hypothesis)

- Let  $\mathcal{M}_{v_1} = \{h_{u_1}^{(t-1)} | u_1 \in \mathcal{N}_1(v_1)\}$  and  $\mathcal{M}_{v_2} = \{h_{u_2}^{(t-1)} | u_2 \in \mathcal{N}_1(v_2)\}$

- At  $t \geq 1$ ,

- As  $h_{v_1}^{(t-1)} = h_{v_2}^{(t-1)}$  and  $\mathcal{M}_{v_1} = \mathcal{M}_{v_2}$ ,

- $h_{v_1}^{(t)} = h_{v_2}^{(t)}$  (irrespective of any aggregation or merge function)

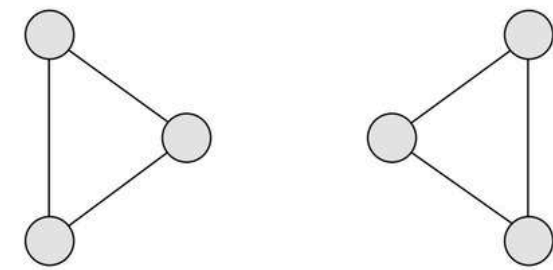
$$a_v^{(t)} = \text{AGGREGATE}^{(t)}\left(\left\{h_u^{(t-1)} | u \in \mathcal{N}_1(v)\right\}\right)$$

$$h_v^{(t)} = \text{MERGE}^{(t)}\left(h_v^{(t-1)}, a_v^{(t)}\right)$$

**Lemma 1.** *The standard GNN maps the nodes of two regular graphs of the same size and degree to the same feature vector.*

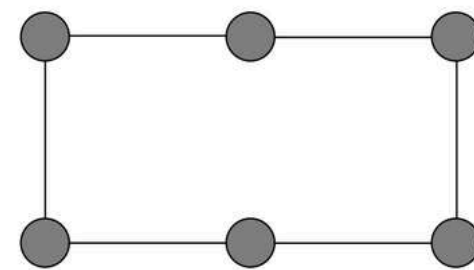
# Limitation of standard GNNs (3)

**Lemma 1.** *The standard GNN maps the nodes of two regular graphs of the same size and degree to the same feature vector.*



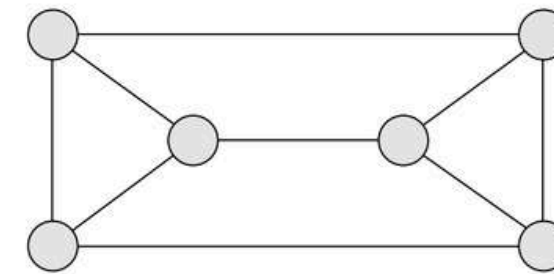
$G_1$

Not connected



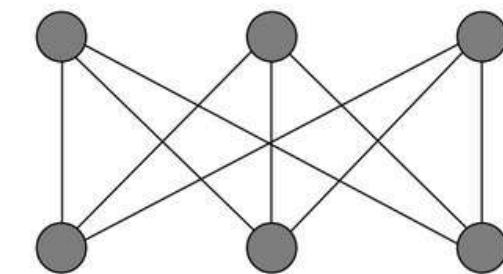
$G_2$

Connected



$G_3$

Not bipartite  
Not triangle-free



$G_4$

Bipartite  
Triangle-free

**Theorem 1.** *The standard GNN cannot identify connectivity, bipartiteness or triangle freeness.*



# Other methods to address these problems

(Cupertino et al., 2017)



Expert Systems with  
Applications

Volume 92, February 2018, Pages 289-303



A scheme for high level data  
classification using random  
walk and network measures

(Silva & Zhao, 2012)

## **A Network-Based High-Level Data Classification Algorithm Using Betweenness Centrality**

Esteban Wilfredo Vilca Zuñiga<sup>1</sup>, Liang Zhao<sup>1</sup>

<sup>1</sup> Dept. of Computing and Mathematics  
FFCLRP-USP  
Ribeirão Preto, Brasil

evilcazu@usp.br, zhao@usp.br

## **Organizational Data Classification Based on the *Importance* Concept of Complex Networks**

Murillo G. Carneiro, *Member, IEEE*, Liang Zhao, *Senior Member, IEEE*

(Carneiro & Zhao, 2017)

Focus on creating features for node classification tasks

# Part 4: K-hop GNNs

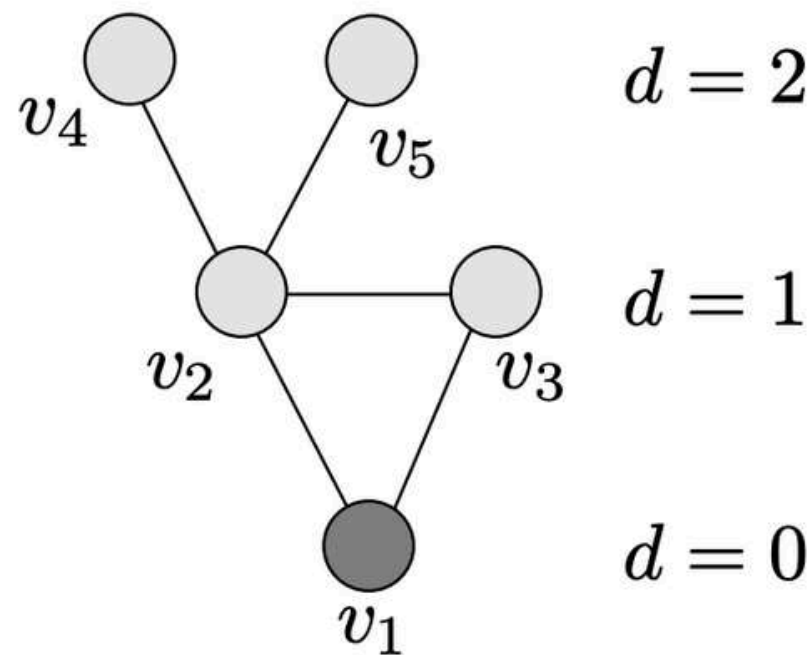
# Overview of k-hop GNNs

- Use information of k-hop neighbours (instead of only direct neighbour)

$$a_v^{(t)} = \text{AGGREGATE}^{(t)} \left( \left\{ h_u^{(t-1)} \mid u \in \mathcal{N}_{\boxed{k}}(v) \right\} \right)$$
$$h_v^{(t)} = \text{MERGE}^{(t)} \left( h_v^{(t-1)}, a_v^{(t)} \right)$$



# Architecture of k-hop GNNs (1)



Intuition:

For every node ( $v_1$ ) to pass through the aggregation layer:

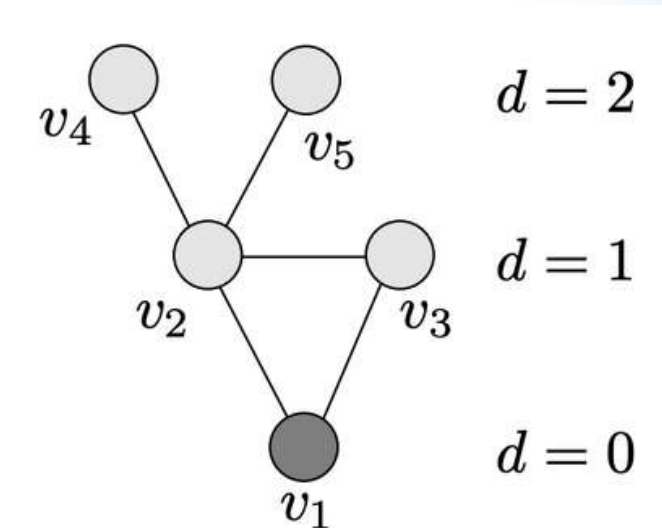
- Each neighbours have internal representation
- Start updating from further nodes (at k-hop)
- Update across previous layer, and then update within current layer

# Architecture of k-hop GNNs (2)

- $G_v^k$ : Subgraph induced by node  $v$  and its  $k$ -hop neighbour
- $\text{UPDATE}(w, S) = \text{MLP}\left(\text{MLP}_1(x_w) + \sum_{u \in S} \text{MLP}_2(x_u)\right)$

$w$ : node;  $S$ : a set of nodes;  $x_u$ : Inner Representation of  $u$

- $R_d(v)$ : The set of nodes exactly at  $k$ -hop from  $v$   
(Ring of nodes of  $v$  at distance  $d$ )



Update across ring of nodes

$$x_u = \text{UPDATE}_{d, \text{across}}^{(t)}(u, \mathcal{B})$$

$$\mathcal{B} = \mathcal{N}_1(u) \cap R_{d+1}(v)$$

Update within a ring of nodes

$$x_u = \text{UPDATE}_{d, \text{within}}^{(t)}(u, \mathcal{D})$$

$$\mathcal{D} = \mathcal{N}_1(u) \cap R_d(v)$$

# Algorithm & Complexity Analysis

---

## Algorithm 1: $k$ -hop GNN

---

**Input:** Graph  $G = (V, E)$ , node features  $\{h_v : v \in V\}$ , number of neighborhood aggregation layers  $T$ , number of hops  $k$

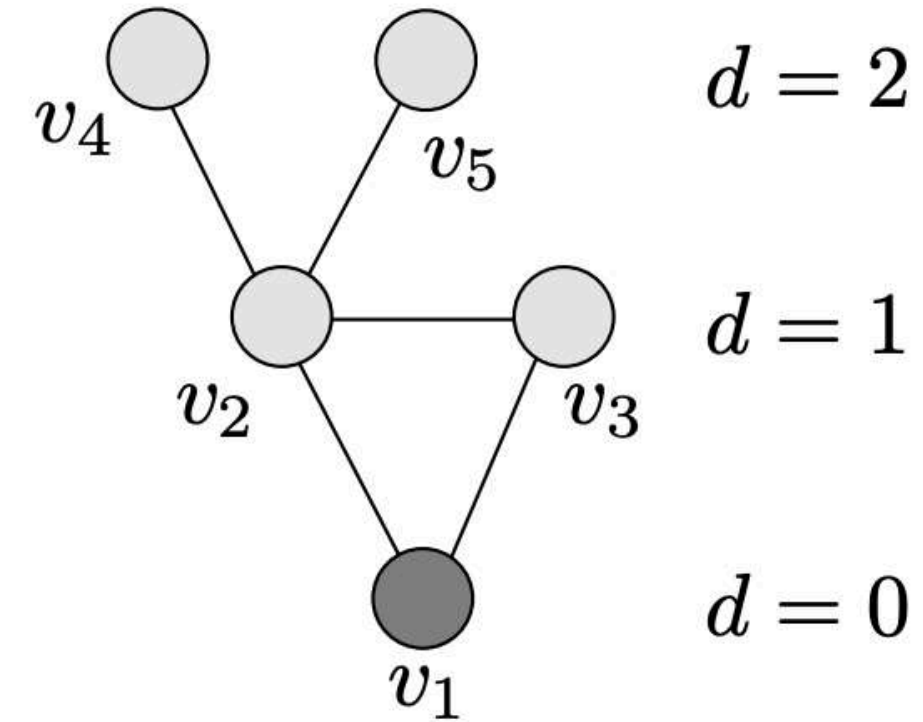
**Output:** Node features  $\{h_v^{(T)} : v \in V\}$

```

1: for  $t \in \{1, \dots, T\}$  do   For every aggregation layer  $t$ 
2:   for  $v \in V$  do   For every vertex  $v$ 
3:     for  $u \in R_k(v)$  do   Start from the furthest layer (Only needs to update within layer)
4:        $\mathcal{D} \leftarrow \mathcal{N}_1(u) \cap R_k(v)$    for  $u=v_4, \{\} \cap \{v_4, v_5\}$ 
5:        $x_u \leftarrow \text{UPDATE}_{k, \text{within}}^{(t)}(u, \mathcal{D})$ 
6:     end for
7:     for  $i \in \{k-1, \dots, 1\}$  do
8:       for  $u \in R_i(v)$  do   for  $i=1, u=v_2$ 
9:          $\mathcal{B} \leftarrow \mathcal{N}_1(u) \cap R_{i+1}(v)$     $\{v_4, v_5, v_3\} \cap \{v_4, v_5\}$ 
10:         $x_u \leftarrow \text{UPDATE}_{i, \text{across}}^{(t)}(u, \mathcal{B})$ 
11:         $\mathcal{D} \leftarrow \mathcal{N}_1(u) \cap R_i(v)$     $\{v_4, v_5, v_3\} \cap \{v_2, v_3\}$ 
12:         $x_u \leftarrow \text{UPDATE}_{i, \text{within}}^{(t)}(u, \mathcal{D})$ 
13:      end for
14:    end for
15:     $h_v^{(t)} = \text{UPDATE}_{0, \text{across}}^{(t)}(v, \mathcal{N}_1(v))$    Update the root node with direct neighbour
16:  end for
17: end for

```

---



	Dense Graph	Sparse Graph
Preprocessing	$\mathcal{O}(nm)$	$\mathcal{O}(n \bar{d}^k)$
Message Passing	$\mathcal{O}(nm)$	$\mathcal{O}(n \bar{d}^k)$

Note: Complexity of GNNs is  $\mathcal{O}(km)$



# Expressiveness of k-hop GNNs

**Theorem 2.** *For the k-hop GNN, there exists a sequence of modules  $UPDATE_{0,across}^{(0)}$ ,  $UPDATE_{1,within}^{(0)}$ ,  $UPDATE_{1,across}^{(0)}$ ,  $\dots$ ,  $UPDATE_{k-1,across}^{(T)}$ ,  $UPDATE_{k,within}^{(T)}$  such that*

- 1. it can identify triangle-freeness for  $k \geq 1$*
- 2. connectivity for  $k > \delta_{min}$  where  $\delta_{min}$  is the minimum of the diameters of the connected components*
- 3. bipartiteness for  $k \geq \frac{l-1}{2}$  where  $l$  is the length of the smallest odd cycle in the graph (if any)*

(Proof in appendix)

# Part 5: Experimental Evaluation

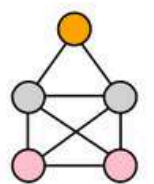
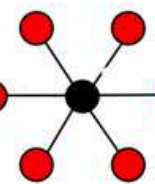
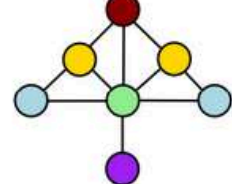
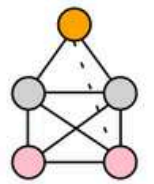
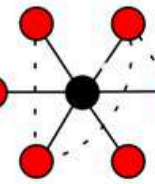
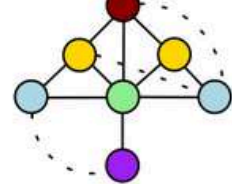
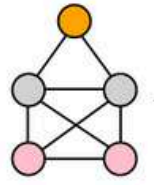
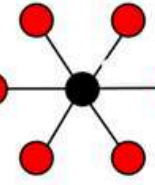
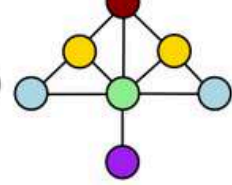
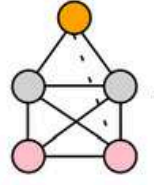
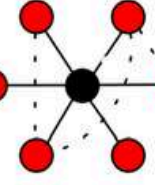
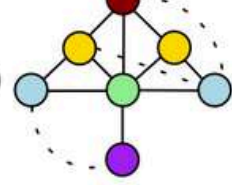
# Overview of the experiment

- 2 tasks: Node Classification and Graph Classification
- Evaluated on synthetic and real-world datasets
- Evaluation metrics: Avg accuracy and F1-score
- Compared with standard and other SOTA GNNs
- Each model was trained using cross-validation
- Representation of unsupervised learning is mapped to prediction using 4-NN



# Synthetic Datasets on Node Classification

- All graphs consist of a cycle of length 40
- 10 instances are added according to following configurations:

CONFIGURATION	SHAPES PLACED ALONG A CYCLE GRAPH
BASIC	 OR  OR 
BASIC PERTURBED	 OR  OR 
VARIED	 AND  AND 
VARIED PERTURBED	 AND  AND 

# Performance on node classification (synthetic)

METHOD	CONFIGURATION							
	BASIC		BASIC PERTURBED		VARIED		VARIED PERTURBED	
	ACCURACY	F1-SCORE	ACCURACY	F1-SCORE	ACCURACY	F1-SCORE	ACCURACY	F1-SCORE
RoLX	<b>1.000</b>	<b>1.000</b>	0.928	0.886	0.998	0.996	0.856	0.768
STRUC2VEC	0.784	0.708	0.703	0.632	0.738	0.592	0.573	0.412
GRAPHWAVE	0.995	0.993	0.906	0.861	0.982	0.965	0.793	0.682
2-GNN	0.997	0.994	0.920	0.876	0.990	0.979	0.852	0.753
3-GNN	0.997	0.994	0.911	0.859	0.993	0.985	0.866	0.775
CHEBNET (K=2)	0.988	0.979	0.866	0.787	0.852	0.732	0.624	0.471
CHEBNET (K=3)	0.992	0.987	0.904	0.850	0.958	0.917	0.758	0.612
ARMA (T=2)	0.996	0.992	0.914	0.861	0.982	0.961	0.839	0.728
ARMA (T=3)	0.997	0.996	0.919	0.872	0.993	0.987	0.850	0.747
2-HOP GNN	<b>1.000</b>	<b>1.000</b>	0.961	<b>0.934</b>	<b>0.999</b>	<b>0.999</b>	0.948	0.910
3-HOP GNN	<b>1.000</b>	<b>1.000</b>	<b>0.962</b>	<b>0.934</b>	0.996	0.993	<b>0.952</b>	<b>0.916</b>

K-hop GNN outperforms all model in every configuration

# Real-world Datasets on Node Classification

## Enron dataset

- E-mail network encoding communication between employees in a company
- Task: predict the job function of an employee (7 classes, e.g., CEO, manager, etc.)
- Nodes: Enron employees; edges: e-mail communication between the employees
- 143 nodes and 2583 edges



# Performance on node classification (real-world)

METHOD	ACCURACY	F1-SCORE
RoLX	0.264	0.154
STRUC2VEC	0.323	0.190
GRAPHWAVE	0.257	0.149
2-GNN	0.357	0.183
3-GNN	0.366	0.195
CHEBNET (K=2)	0.342	0.179
CHEBNET (K=3)	0.360	0.191
ARMA (T=2)	0.374	0.192
ARMA (T=3)	<b>0.376</b>	0.190
2-HOP GNN	0.366	<b>0.198</b>
3-HOP GNN	0.327	0.171

- Supervised > Unsupervised
- 2-hop > 3-hop (Is k domain/topic specific?)

# Synthetic Datasets on Graph Classification

- 3 datasets to test triangle-freeness, bipartiteness and connectivity respectively
- Each consists of 800 4-regular graphs of 60 nodes
- Classes are evenly distributed

# Performance on graph classification (synthetic)

METHOD	PROPERTY		
	CONNECTIVITY	BIPARTITENESS	TRIANGLE FREENESS
2-GNN	55.00 $\pm$ 5.30	53.78 $\pm$ 2.61	51.87 $\pm$ 7.43
3-GNN	56.20 $\pm$ 2.28	58.13 $\pm$ 2.10	55.90 $\pm$ 4.44
CHEBNET (K=2)	56.37 $\pm$ 7.76	50.33 $\pm$ 1.20	53.12 $\pm$ 6.35
CHEBNET (K=3)	57.62 $\pm$ 3.84	51.98 $\pm$ 3.56	54.75 $\pm$ 7.14
ARMA (T=2)	55.55 $\pm$ 5.59	54.50 $\pm$ 4.61	53.00 $\pm$ 3.18
ARMA (T=3)	55.63 $\pm$ 5.69	53.92 $\pm$ 3.22	54.25 $\pm$ 5.80
2-HOP GNN	81.24 $\pm$ 5.22	84.69 $\pm$ 1.74	<b>84.06</b> $\pm$ 2.12
3-HOP GNN	<b>94.77</b> $\pm$ 3.41	<b>91.12</b> $\pm$ 2.76	82.53 $\pm$ 5.33

- Standard GNNs cannot identify the properties
- K-hop GNN outperforms all models



# Real-world Datasets on Graph Classification

- Datasets for bioinformatics and chemoinformatics:
  - (1) MUTAG, (2) PROTEINS, (3) NCI1
- Datasets for social interaction networks
  - (1) IMDB-BINARY, (2) IMDB-MULTI

DATASET	STATISTICS					
	#GRAPHS	#CLASSES	MAX CLASS IMBALANCE	AVG. #NODES	AVG. #EDGES	LABELS (NUM.)
MUTAG	188	2	1 : 1.98	17.93	19.79	+ (7)
PROTEINS	1,113	2	1 : 1.47	39.06	72.82	+ (3)
NCI1	4,110	2	1 : 1	29.87	32.30	+ (37)
IMDB-BINARY	1,000	2	1 : 1	19.77	96.53	–
IMDB-MULTI	1,500	3	1 : 1	13.00	65.94	–

# Performance on graph classification (real-world)

METHOD \ DATASET	MUTAG	PROTEINS	NCI1	IMDB BINARY	IMDB MULTI	AVERAGE RANK
GK	69.97 ( $\pm$ 2.22)	71.23 ( $\pm$ 0.38)	65.47 ( $\pm$ 0.14)	60.33 ( $\pm$ 0.25)	36.53 ( $\pm$ 0.93)	16.8
SP	84.03 ( $\pm$ 1.49)	75.36 ( $\pm$ 0.61)	72.85 ( $\pm$ 0.24)	60.21 ( $\pm$ 0.58)	39.62 ( $\pm$ 0.57)	12.8
WL	83.63 ( $\pm$ 1.57)	73.12 ( $\pm$ 0.52)	84.42 ( $\pm$ 0.25)	73.36 ( $\pm$ 0.38)	<b>51.06</b> ( $\pm$ 0.47)	6.8
WL-OA	86.63 ( $\pm$ 1.49)	75.35 ( $\pm$ 0.45)	<b>85.74</b> ( $\pm$ 0.37)	73.61 ( $\pm$ 0.60)	50.48 ( $\pm$ 0.33)	<b>3.0</b>
GS-SVM	83.57 ( $\pm$ 6.75)	74.11 ( $\pm$ 4.02)	79.14 ( $\pm$ 1.28)	71.20 ( $\pm$ 3.25)	48.73 ( $\pm$ 2.32)	10.8
2-GNN	85.92 ( $\pm$ 2.19)	75.24 ( $\pm$ 0.45)	76.32 ( $\pm$ 0.41)	71.40 ( $\pm$ 0.74)	47.73 ( $\pm$ 0.86)	8.8
3-GNN	85.74 ( $\pm$ 1.48)	74.59 ( $\pm$ 0.71)	79.62 ( $\pm$ 0.45)	71.60 ( $\pm$ 0.84)	47.33 ( $\pm$ 1.01)	9.4
CHEBYNET (K=2)	85.33 ( $\pm$ 1.42)	74.72 ( $\pm$ 0.97)	78.97 ( $\pm$ 0.35)	71.08 ( $\pm$ 0.51)	47.08 ( $\pm$ 0.60)	10.6
CHEBYNET (K=3)	82.49 ( $\pm$ 1.52)	74.81 ( $\pm$ 0.82)	81.01 ( $\pm$ 0.39)	70.90 ( $\pm$ 0.73)	46.66 ( $\pm$ 0.59)	10.8
ARMA (T=2)	82.98 ( $\pm$ 1.90)	74.84 ( $\pm$ 0.59)	80.83 ( $\pm$ 0.42)	70.62 ( $\pm$ 0.95)	46.10 ( $\pm$ 0.82)	11.2
ARMA (T=3)	81.52 ( $\pm$ 1.22)	74.74 ( $\pm$ 0.67)	81.34 ( $\pm$ 0.38)	70.52 ( $\pm$ 0.71)	46.12 ( $\pm$ 0.98)	11.6
PATCHYSAN ( $k = 10$ )	<b>88.95</b> ( $\pm$ 4.37)	75.00 ( $\pm$ 2.51)	76.34 ( $\pm$ 1.68)	71.00 ( $\pm$ 2.29)	45.23 ( $\pm$ 2.84)	9.6
DGCNN	85.83 ( $\pm$ 1.66)	75.54 ( $\pm$ 0.94)	74.44 ( $\pm$ 0.47)	70.03 ( $\pm$ 0.86)	47.83 ( $\pm$ 0.85)	9.6
CAPSGNN	86.67 ( $\pm$ 6.88)	<b>76.28</b> ( $\pm$ 3.63)	78.35 ( $\pm$ 1.55)	73.10 ( $\pm$ 4.83)	50.27 ( $\pm$ 2.65)	<b>5.0</b>
1-2-3-GNN	86.1	75.5	76.2	<b>74.2</b>	49.5	6.0
2-HOP GNN	87.93 ( $\pm$ 1.22)	75.03 ( $\pm$ 0.42)	79.31 ( $\pm$ 0.57)	73.33 ( $\pm$ 0.30)	49.79 ( $\pm$ 0.25)	<b>5.4</b>
3-HOP GNN	87.56 ( $\pm$ 0.72)	75.28 ( $\pm$ 0.36)	80.61 ( $\pm$ 0.34)	-	-	<b>4.8</b>

- K-hop GNN perform slightly lower than the best model in every task
- K-hop GNN > Standard GNN
- 2 vs 3 hops do not have much difference

# Part 6: Conclusion



# Summary

- Limitation of standard GNNs
  - Unable to identify connectivity, bipartiteness and triangle freeness
- New architecture: k-hop GNNs
  - High accuracy in identifying the 3 properties
- Performance evaluation and comparison



# Strengths

- Show k-hop GNNs is better in a all-rounded way
- The experimental setup is fair
- Proposed an architecture that is easy to be implemented

# Weaknesses

- Only evaluate on classification tasks
- Better evaluation on complexity (FLOPs)
- Real-life GNNs also take in other level of info (edge, global)
- Mapping 4-NN to map representation from unsupervised model is not fair

# Future Research

- Necessity to use k-hop GNNs when we have edge and global level features?
- How powerful in identifying other graph features?  
(Exp: Tree, planer graph)
- Best way to implement k-hop GNNs?
  - More efficient way? Better representation?
  - Necessary to update inner representation of all neighbour twice using MLPs?
  - Is it hard to train when the network is big and complex?

# Reference

Carneiro, M. G., & Zhao, L. (2017). Organizational data classification based on the importance concept of complex networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8), 3361–3373. <https://doi.org/10.1109/tnnls.2017.2726082>

Cupertino, T. H., Carneiro, M. G., Zheng, Q., Zhang, J., & Zhao, L. (2017). A scheme for high level data classification using random walk and network measures. *Expert Systems With Applications*, 92, 289–303. <https://doi.org/10.1016/j.eswa.2017.09.014>

Kriege, N. M., Morris, C., Rey, A., & Sohler, C. (2018). A Property Testing Framework for the Theoretical Expressivity of Graph Kernels. *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2348–2354. <https://doi.org/10.24963/ijcai.2018/325>

Nikolentzos, G., Dasoulas, G., & Vazirgiannis, M. (2020). k-hop graph neural networks. *Neural Networks*, 130, 195–205. <https://doi.org/10.1016/j.neunet.2020.07.008>

Sanchez-Lengeling, B., Reif, E., Pearce, A., & Wiltchko, A. (2021). A gentle introduction to graph neural networks. *Distill*, 6(8). <https://doi.org/10.23915/distill.00033>

Silva, T. C., & Zhao, N. L. (2012). Network-Based high level data classification. *IEEE Transactions on Neural Networks and Learning Systems*, 23(6), 954–970. <https://doi.org/10.1109/tnnls.2012.2195027>



# ***THANK YOU FOR LISTENING!***

Presented by  
**Lee Guan Bo Ambrose**  
BASc (Applied AI)



UNIVERSITY OF  
**WATERLOO**

DAVID R. CHERITON SCHOOL  
OF COMPUTER SCIENCE



SCHOOL OF  
**COMPUTING &  
DATA SCIENCE**  
The University of Hong Kong

# *Part -1: Appendix*

# Proof for theorem 2 (0)

We assume that the feature vectors of the nodes come from a countable set. This set may correspond to a subset of an uncountable set such as  $\mathbb{R}^d$ . Furthermore, the feature vectors of a set of nodes form a multiset (i. e., since some nodes may have identical feature vectors).

We will show that if a graph has one of the three considered properties (i. e., triangle-freeness, bipartiteness, and connectivity), some of its nodes can be mapped to different feature vectors compared to the nodes of a graph that does not have the property. Then, by applying an injective readout function, the two graphs can also be mapped to different feature vectors.

For simplicity of presentation, we will assume that the proposed model consists of a single neighborhood aggregation layer. The same results also hold for multiple neighborhood aggregation layers. We first show that the aggregation scheme that our model employs can represent universal functions over the pairs of a node and the multiset of its neighbors. The following Lemma generalizes the setting in [39].



# Proof for theorem 2 (1)

**Lemma 2.** *Assume  $\mathcal{X}$  is countable, and let  $r \in \mathbb{N}$ . There exist functions  $f : \mathcal{X} \rightarrow \mathbb{R}^d$  and  $f' : \mathcal{X} \rightarrow \mathbb{R}^d$ , such that  $h_i(c, X) = f(c) + \sum_{x \in X} f'(x)$  is unique for each  $i \in \{0, \dots, r\}$  and each pair  $(c, X)$ , where  $c \in \mathcal{X}$  and  $X \subset \mathcal{X}$  is a finite multiset. Moreover, any function  $g_i$  over such pairs can be decomposed as  $g_i(c, X) = \phi(f(c) + \sum_{x \in X} f'(x))$  for some function  $\phi$ .*

*Proof.* We first show that there exists a mapping  $f'$  such that  $\sum_{x \in X} f'(x)$  is unique for each finite multiset  $X$ . Because  $\mathcal{X}$  is countable, there exists a mapping  $Z : \mathcal{X} \rightarrow \mathbb{N}$  from  $x \in \mathcal{X}$  to natural numbers. Because the multisets  $X$  are finite, there exists a number  $N \in \mathbb{N}$  such that  $|X| < N$  for all  $X$ . Then, an example of such  $f'$  is  $f'(x) = N^{-Z(x)}$ . The sum of the above function for all finite multisets  $X$  takes values less than 1, i.e.,  $\sum_{x \in X} f'(x) < 1$ . Hence, if we also set  $f(x) = Z(x) + (r - i)|\mathcal{X}|$ , then it holds that  $h_i(c, X) = f(c) + \sum_{x \in X} f'(x)$  is an injective function over pairs of elements and multisets, and is also unique for each  $i \in \{0, \dots, r\}$ .

For any function  $g_i$  over the pairs  $(c, X)$ , we can construct such  $\phi$  for the desired decomposition by letting  $g_i(c, X) = \phi(f(c) + \sum_{x \in X} f'(x))$ . Note that such  $\phi$  is well-defined because  $h_i(c, X) = f(c) + \sum_{x \in X} f'(x)$  is injective.  $\square$



# Proof for theorem 2 (2)

In our setting, the UPDATE modules correspond to  $g_i(c, X)$  functions. These modules use multi-layer perceptrons (MLPs) to model and learn  $f, f'$  and  $\phi$  in the above Lemma, thanks to the universal approximation theorem [18]. Note that given two nodes  $v, v'$ , if a node  $u \in \mathcal{N}_k(v)$  obtains a representation that is never obtained by any node  $u' \in \mathcal{N}_k(v')$ , then based on the above Lemma, there exist UPDATE modules such that the root nodes  $v, v'$  are assigned different representations. Hence, for all three properties, it is sufficient to show that at some point of the algorithm, a node of the graph that satisfies the property can obtain a representation that is never obtained by any node of a graph that does not satisfy the property.

# Proof for theorem 2 (3)

**Triangle-freeness** If a graph is not triangle-free, then there exist at least three nodes whose 1-hop neighborhoods contain a triangle. Let  $v$  be such a node. Then, clearly there are at least two nodes  $u \in R_1(v)$  which are connected to each other by an edge. The representations of these nodes are updated as follows:  $x_u = \text{UPDATE}_{1, \text{within}}^{(0)}(u, \mathcal{D})$ . On the other hand, no such update takes place in the case of triangle-free graphs since  $\mathcal{D} = \emptyset$ . Hence, based on Lemma 2, the above  $\text{UPDATE}_{1, \text{within}}^{(0)}$  module can generate different representations for the nodes that participate in a triangle from the representations of the nodes of the neighborhood subgraph of each node of a triangle-free graph.

**Connectivity** Let  $G$  be a disconnected graph and  $C$  its component which has the minimum diameter  $\delta_{\min}$ . Then, for an arbitrary node  $v$  of component  $C$ , it holds that  $R_i(v) = \emptyset$  for all  $i > \delta_{\min}$ . On the other hand, if the graph is connected, for some node  $v$ , it holds that  $|R_i(v)| > 0$  for all  $i \leq \delta$  where  $\delta > \delta_{\min}$  is the diameter of the connected graph. Hence, the representations of the nodes  $u \in R_i(v)$  and  $u' \in R_{i-1}(v)$  are updated as  $x_u = \text{UPDATE}_{i, \text{within}}^{(0)}(u, \mathcal{D})$  and  $x_{u'} = \text{UPDATE}_{i-1, \text{across}}^{(0)}(u', \mathcal{D}')$ , respectively. Based on Lemma 2, the above two UPDATE modules can generate different representations for the nodes of a neighborhood subgraph of a disconnected graph compared to those of the nodes of the neighborhood subgraph of a connected graph.



# Proof for theorem 2 (4)

**Bipartiteness** It is well-known that a graph is bipartite if and only if it does not contain an odd cycle. If  $G$  is bipartite and  $l$  is the length of the smallest odd cycle in  $G$ , then the  $k$ -hop neighborhood subgraphs ( $k \geq \frac{l-1}{2}$ ) of more than one nodes contain a cycle of odd length. According to Lemma 3, the  $k$ -hop neighborhood subgraph of a node  $v$  contains a cycle of odd length if and only if the shortest path lengths from two adjacent nodes  $u, w \in \mathcal{N}_k(v)$  to  $v$  are identical. In other words, there exist two nodes both at the same level  $i$  of the  $k$ -hop neighborhood subgraph of node  $v$  that are connected to each other with an edge. During the

process of updating the representation of the root  $v$ , the feature vectors of these nodes are also updated as follows:  $x_u = \text{UPDATE}_{i, \text{within}}^{(0)}(u, \mathcal{D})$ . This update does not take place in the case of a non-bipartite graph since  $\mathcal{D} = \emptyset$  for all nodes of all neighborhood subgraphs. Based on Lemma 2, these nodes can obtain different representations from all the representations of the nodes of a neighborhood subgraph extracted from a bipartite graph.

**Lemma 3.** *Let  $G_v^k$  be the  $k$ -hop neighborhood subgraph of a node  $v$ . Then,  $G_v^k$  contains a cycle of odd length if and only if the shortest path lengths from two adjacent nodes  $u, w \in \mathcal{N}_k(v)$  to  $v$  are identical.*

*Proof.* Let  $u, w$  be two vertices such that  $u, w \in \mathcal{N}_k(v)$ . Assume that the shortest path lengths between each of these two vertices and the root  $v$  are identical and equal to  $d \in \mathbb{N}$  such that  $d \leq k$ . If  $u$  and  $w$  are connected by an edge, then  $G_v^k$  contains a cycle of length  $2d + 1$  which is clearly an odd number. This proves the first statement. For the second statement, assume that  $G_v^k$  contains a cycle of odd length and there is no edge between two vertices whose shortest path lengths from the root  $v$  are identical. Then, from all the nodes of the cycle, there is a single node  $u$  such that the shortest path distance from the root  $v$  to  $u$  is maximum, and another node  $w$  such that the shortest path distance from the root  $v$  to  $w$  is minimum ( $w$  could correspond to the root itself). Since this is a cycle, there are two paths from  $w$  to  $u$  of length  $d$ . Hence, the length of the cycle is  $2d$  which is an even number, contradicting the assumption.  $\square$