Power and Limitations of Graph Neural Networks

Seminar for Deep Neural Networks

Johannes Weidenfeller

Supervisor
Karolis Martinkus

8 March 2022

Content

Graph Neural Networks

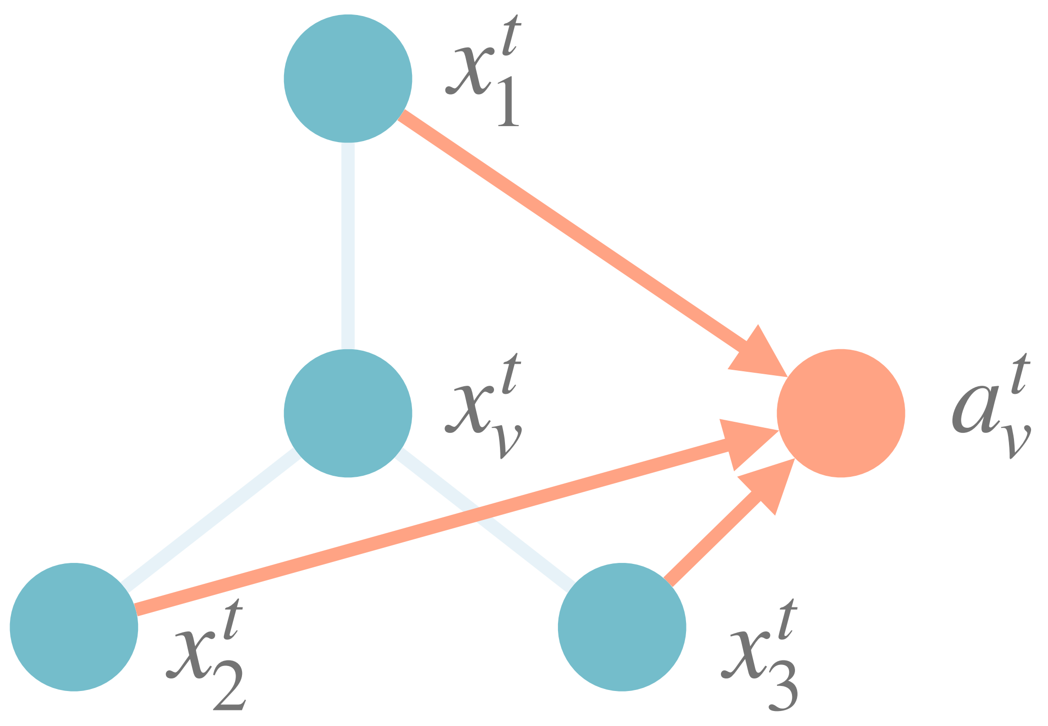
Graph Neural Networks

Basic definition

- Input: Graph $G = (V, E)$ with graph labels x_v for each vertex $v \in V$
- Each layer consists of two steps[1]

1) Aggregate features of neighboring vertices

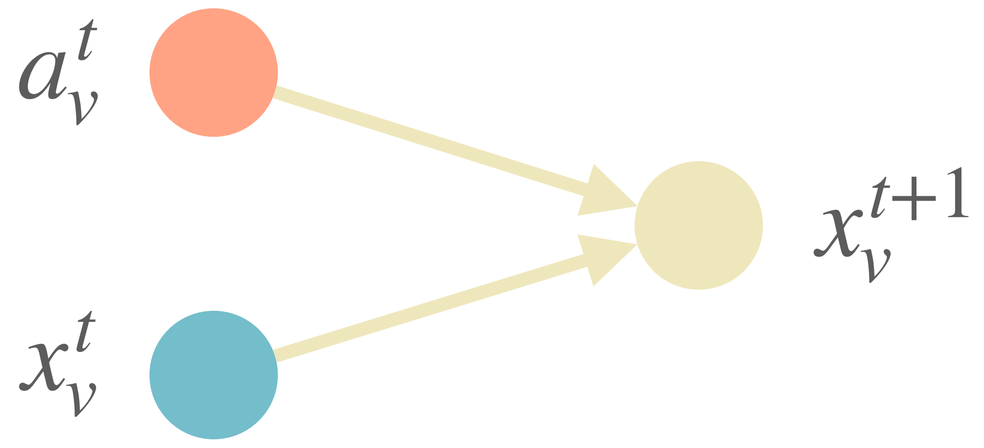
$$a_v^t = AGGREGATE^t \left(\{ x_u^t \,|\, u \in \mathcal{N}(v) \} \right)$$



Examples: Sum, Mean, Max, MLPs

2) Combine aggregate with current vertex label

$$x_v^{t+1} = COMBINE^t \left(x_v^t, a_v^t \right)$$



Examples: Concatenation + Linear Mapping

Graph Neural Networks

Layer function

- We can combine the aggregate and combine functions to a single layer function f_θ

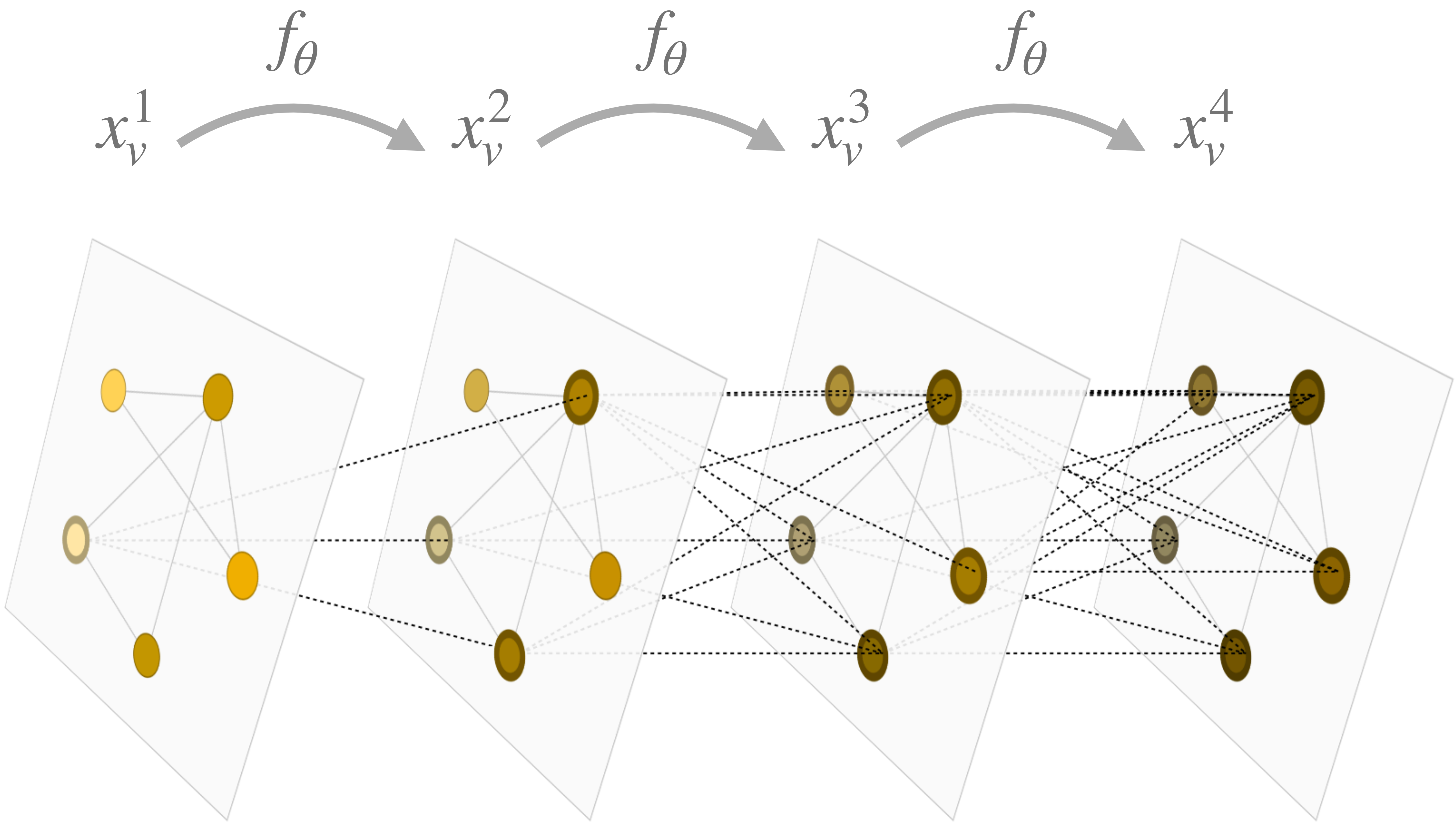
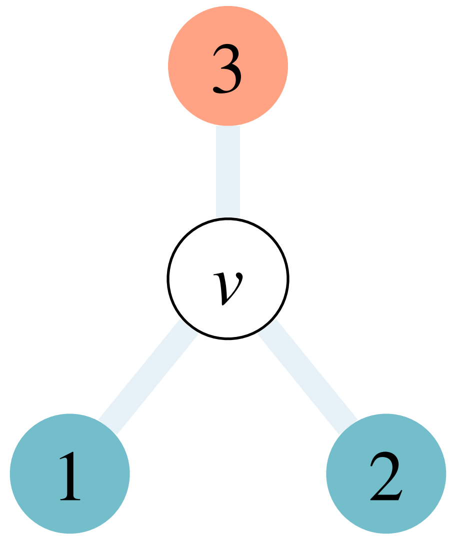
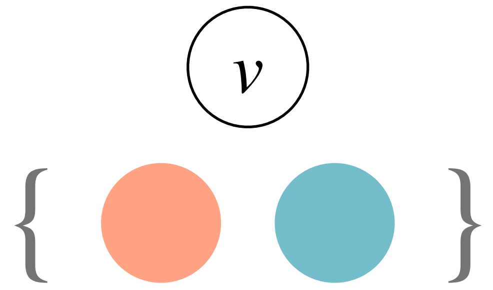
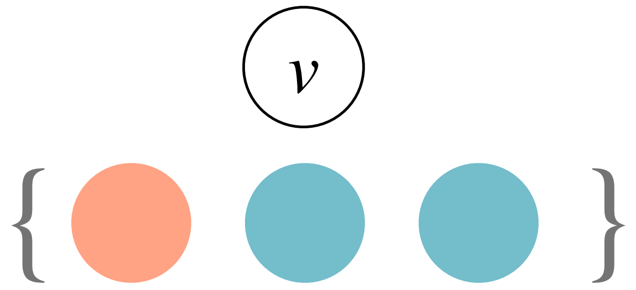
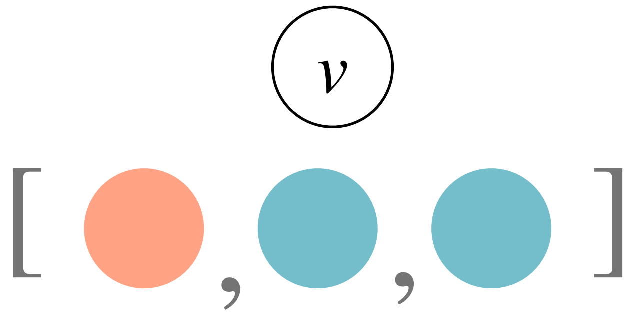
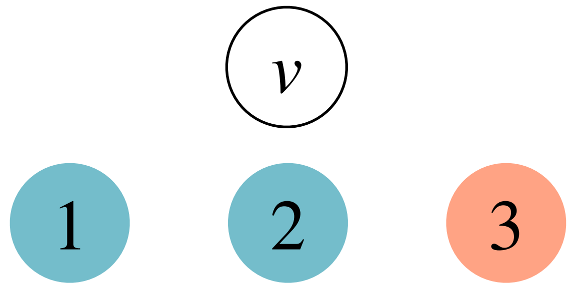


Figure 1: Propagation of information in a graph neural network

Graph Neural Networks

Classification

- Depending on layer function we can distinguish between different GNN classes with different computational complexity[2]

Model	Set broadcasting GNNs	Multiset broadcasting GNNs	Vector-vector consistent GNNs	GNNs with access to unique vertex IDs
				
Input	Set	Multiset	Multiset and port numbering	Unique vertex IDs

Graph Neural Networks

Readout function

- Often, we are interested in graph level classification/regression tasks

- GNNs can be extended through a $READOUT$ function that combines features from all nodes

$$x_G = READOUT\left(\{x_v^T \mid v \in V\}\right)$$

(where T denotes the index of the last layer)

- Should be permutation invariant

- Examples: Summation, Mean/Max-Pooling

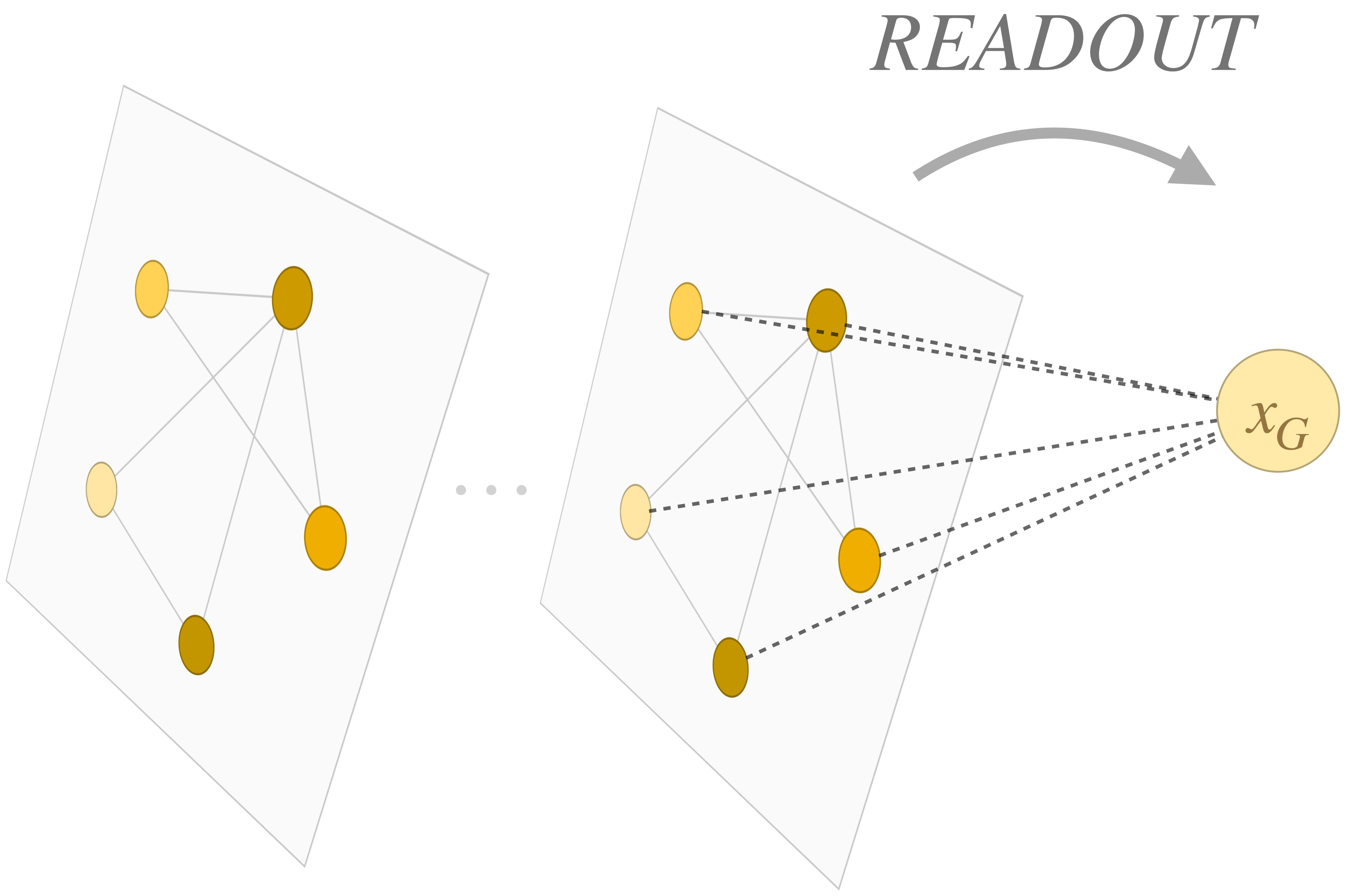


Figure 2: $READOUT$ function in a graph neural network

Graph Neural Networks

Depth and Width

Definition 1

The depth d of a Graph Neural Networks is equal to its number of layers.

Definition 2

The width w of a Graph Neural Network is equal to the largest dimension of x_v^t for any vertex v and layer t

$$w = \max_{v \in V} \max_{t \in \{0,\ldots,d\}} \dim(x_v^t) .$$

- The depth and width of a GNN play a crucial role in its computational power

The Weisfeiler Leman Isomorphism Test

Graph isomorphism

Definition

Two labeled graphs $G = (V, E, X)$ and $G' = (V', E', X')$ are isomorphic if there exists a bijection $f : V \to V'$, such that

i) $\big(f(u), f(v)\big) \in E'$ for all $(u, v) \in E$

$\big(f^{-1}(u'), f^{-1}(v')\big) \in E$ for all $(u', v') \in E'$

ii) $x'_{f(v)} = x_v$ for all $v \in V$

- In unlabelled case we can omit labels, or set $x_v = 0$ for all vertices $v \in V$
- Unknown whether it is solvable in polynomial time

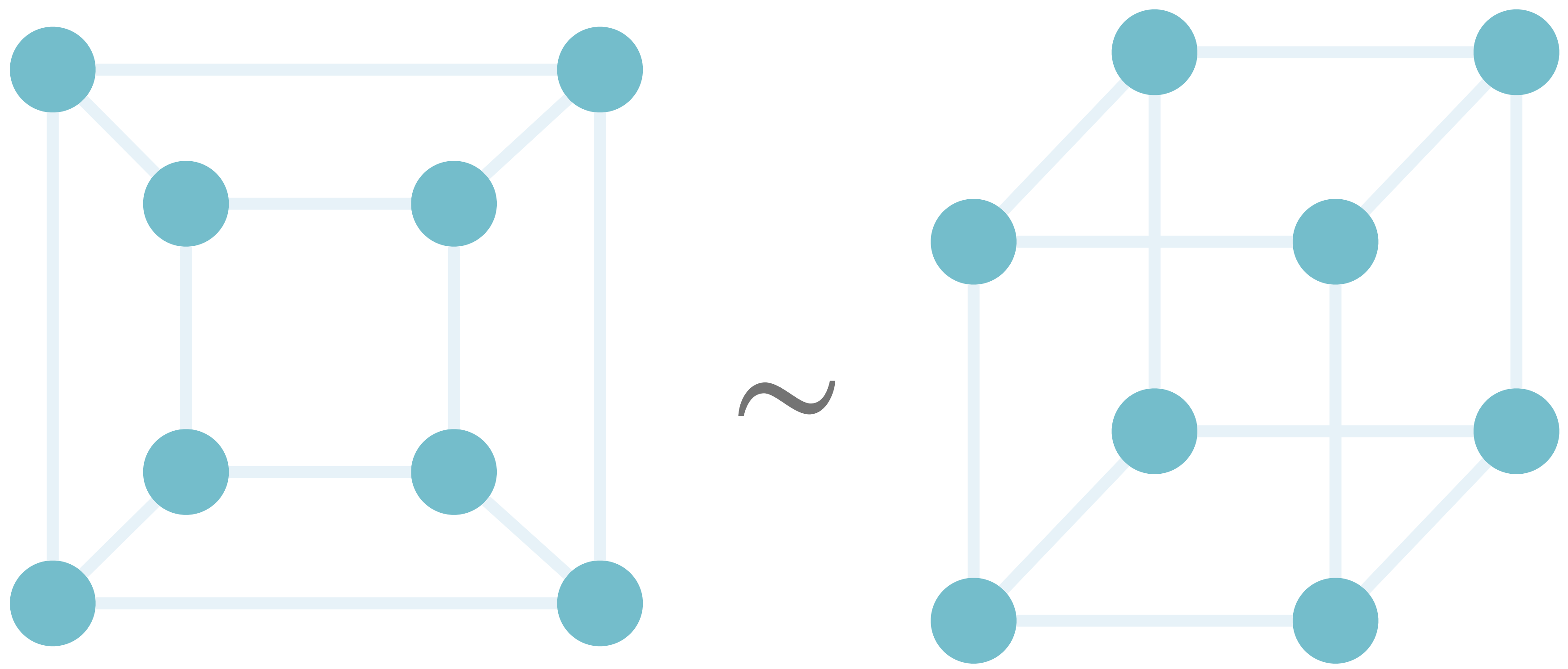


Figure 3: Two (unlabelled) isomorphic graphs

Figure 4: Two (labelled) non-isomorphic graphs

WL Isomorphism Test

Overview

- Algorithm for solving the graph isomorphism problem
- Idea: Iteratively reduce graphs to canonical forms that coincide if graphs are isomorphic
- If canonical forms differ, graphs are non-isomorphic
- In each step t, assign to every node i a label x_i^t

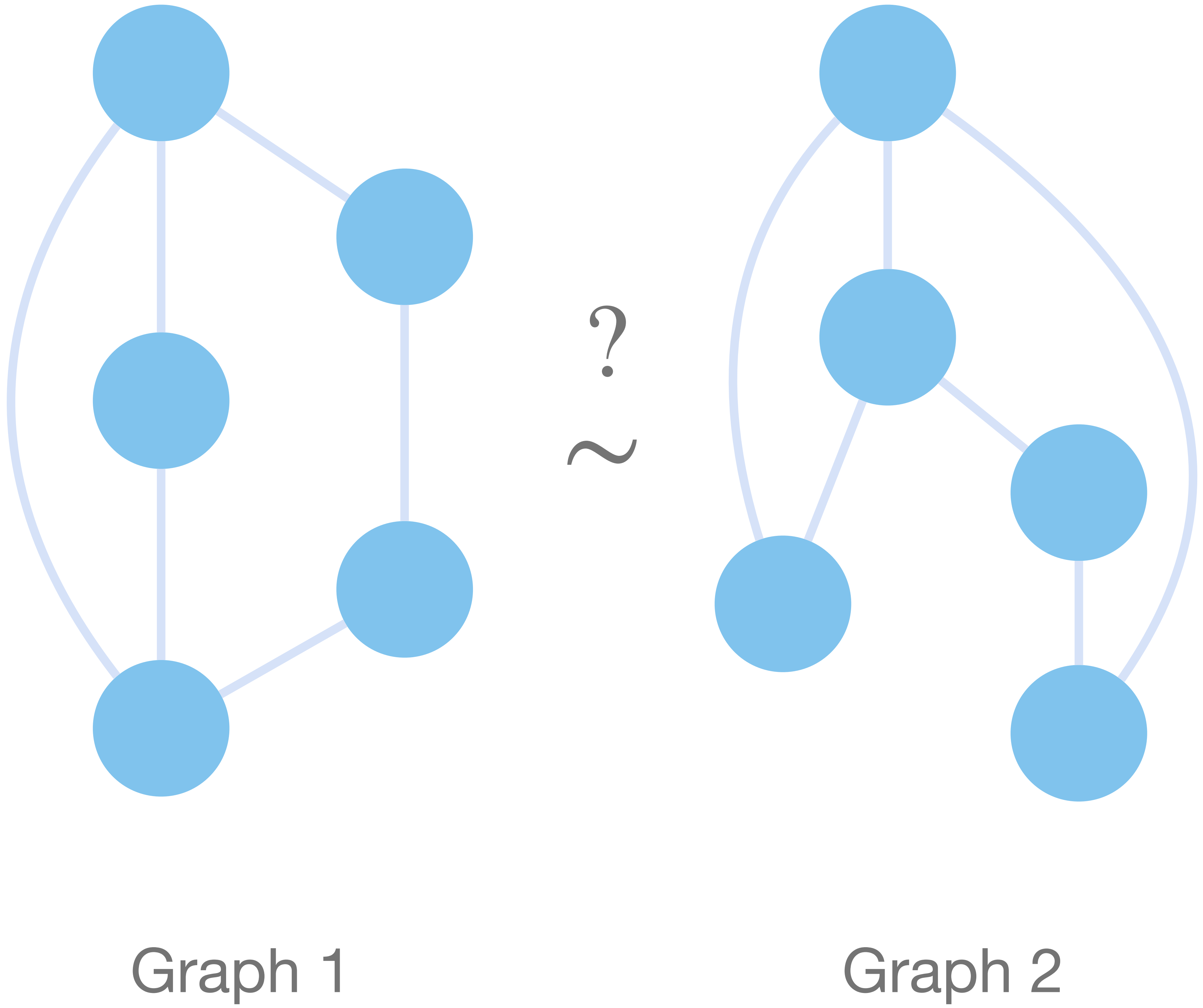


Figure 5: The Weisfeiler-Lehman Isomorphism Algorithm

WL Isomorphism Test

Algorithm[3]

- Initialization: Set node features x_v^0 to original graph labels

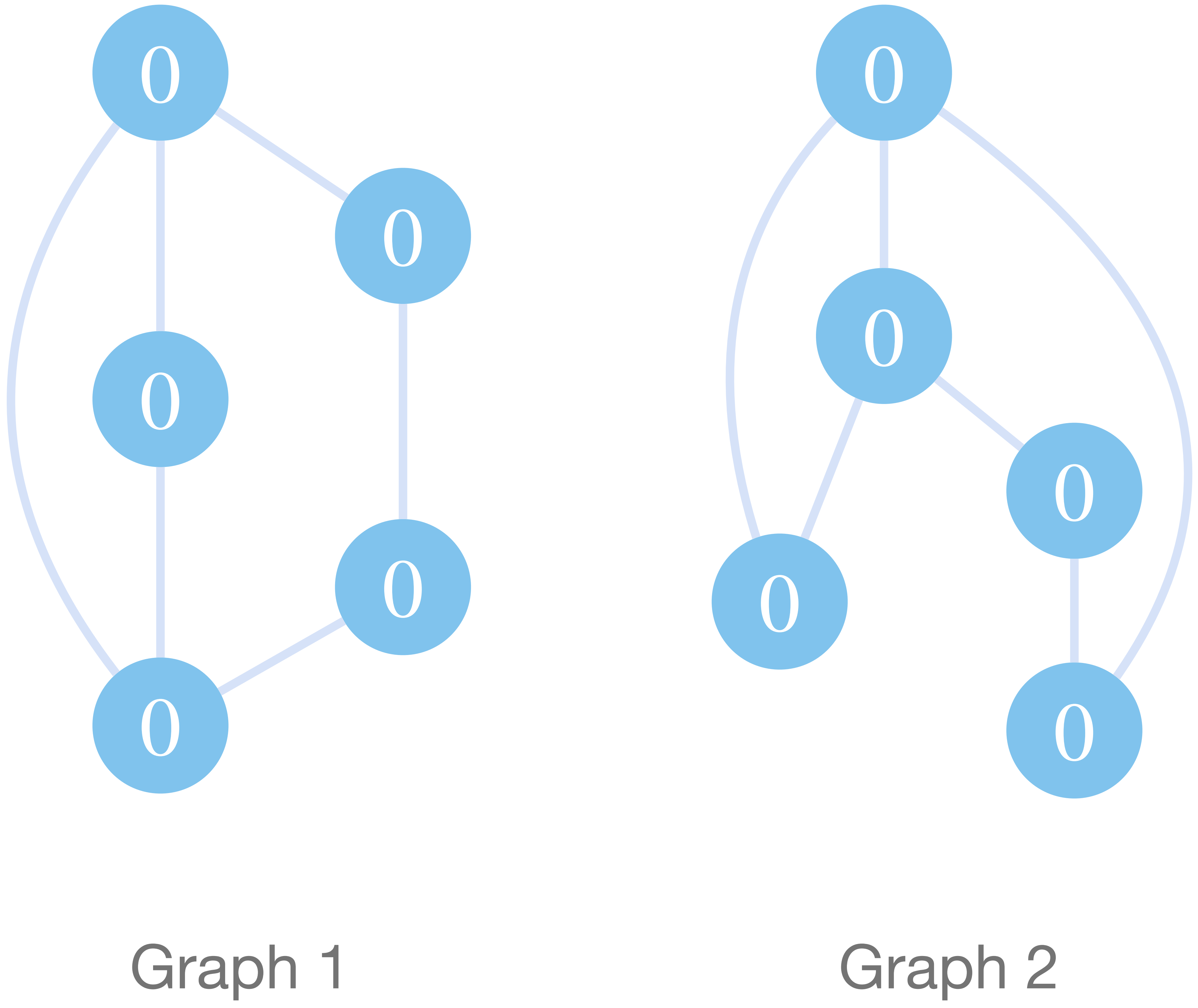


Figure 5: The Weisfeiler-Lehman Isomorphism Algorithm

WL Isomorphism Test

Algorithm[3]

- Initialization: Set node features x_v^0 to original graph labels

- For $t = 0, \ldots, n-1$, repeat

  - For each node v form a multi set S_v^t of the labels of all neighbors

$$S_v^t = \{x_u^t \,|\, u \in \mathcal{N}(v)\}$$

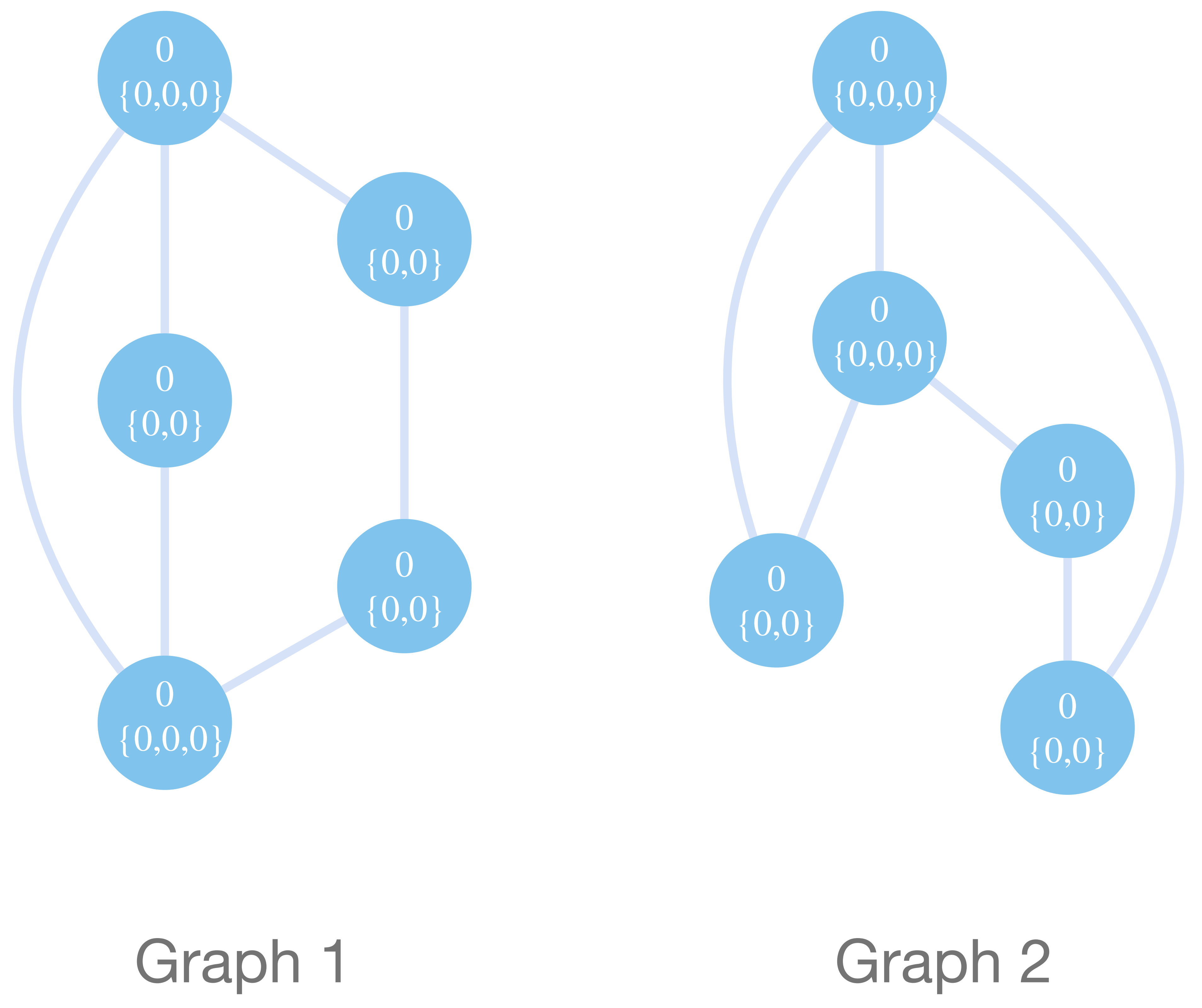


Figure 5: The Weisfeiler-Lehman Isomorphism Algorithm

WL Isomorphism Test

Algorithm[3]

- Initialization: Set node features x_v^0 to original graph labels

- For $t = 0,\ldots,n-1$, repeat
  - For each node v form a multi set S_v^t of the labels of all neighbors

$$S_v^t = \{x_u^t \,|\, u \in \mathcal{N}(v)\}$$

  - Map each pair of label x_v^t and multi set S_v^t to a new label x_v^{t+1} (e.g. via a hash function)

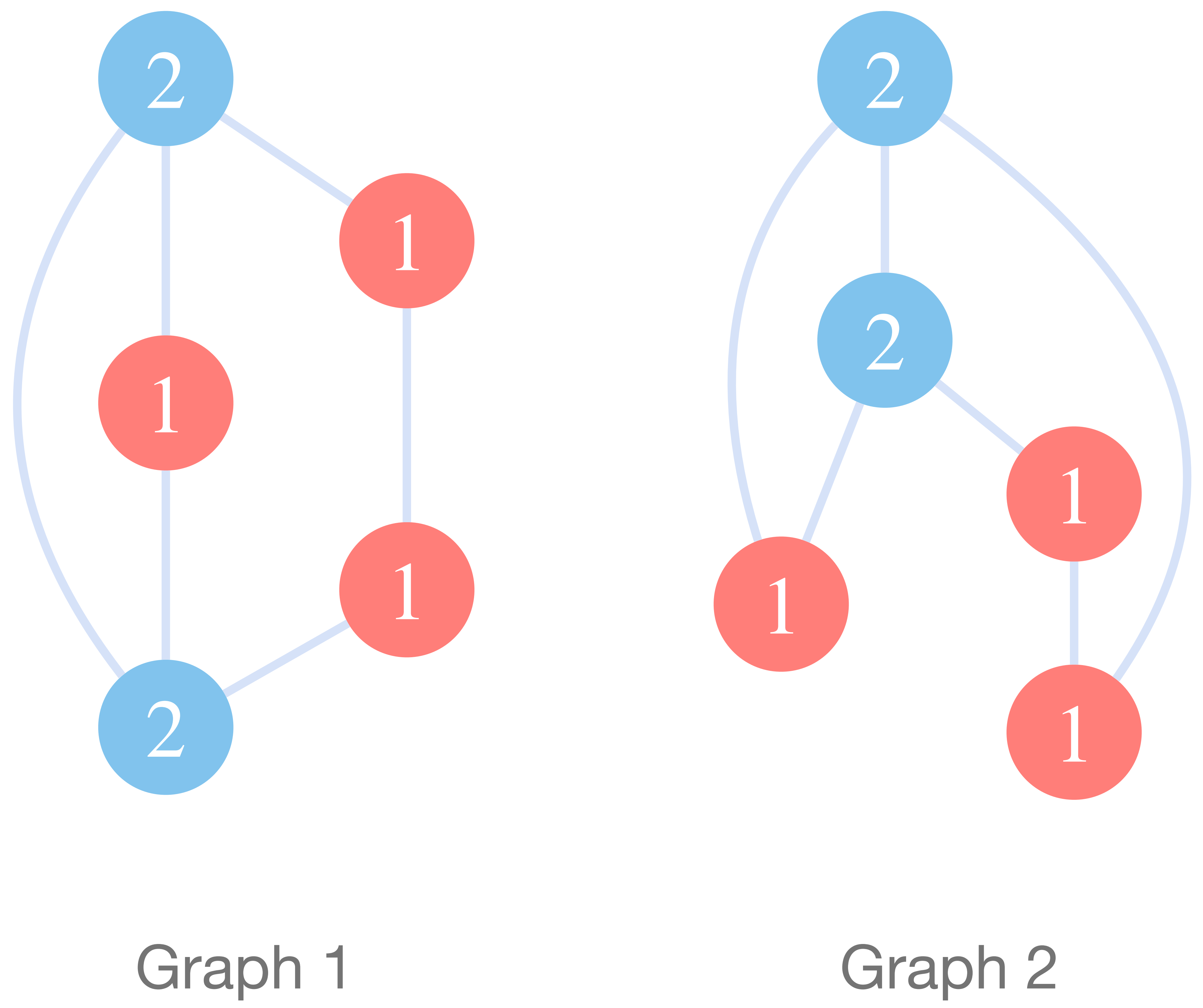


Figure 5: The Weisfeiler-Lehman Isomorphism Algorithm

WL Isomorphism Test

Algorithm[3]

- Initialization: Set node features x_v^0 to original graph labels

- For $t = 0, \ldots, n-1$, repeat

  - For each node v form a multi set S_v^t of the labels of all neighbors

$$S_v^t = \{x_u^t \,|\, u \in \mathcal{N}(v)\}$$

  - Map each pair of label x_v^t and multi set S_v^t to a new label x_v^{t+1} (e.g. via a hash function)

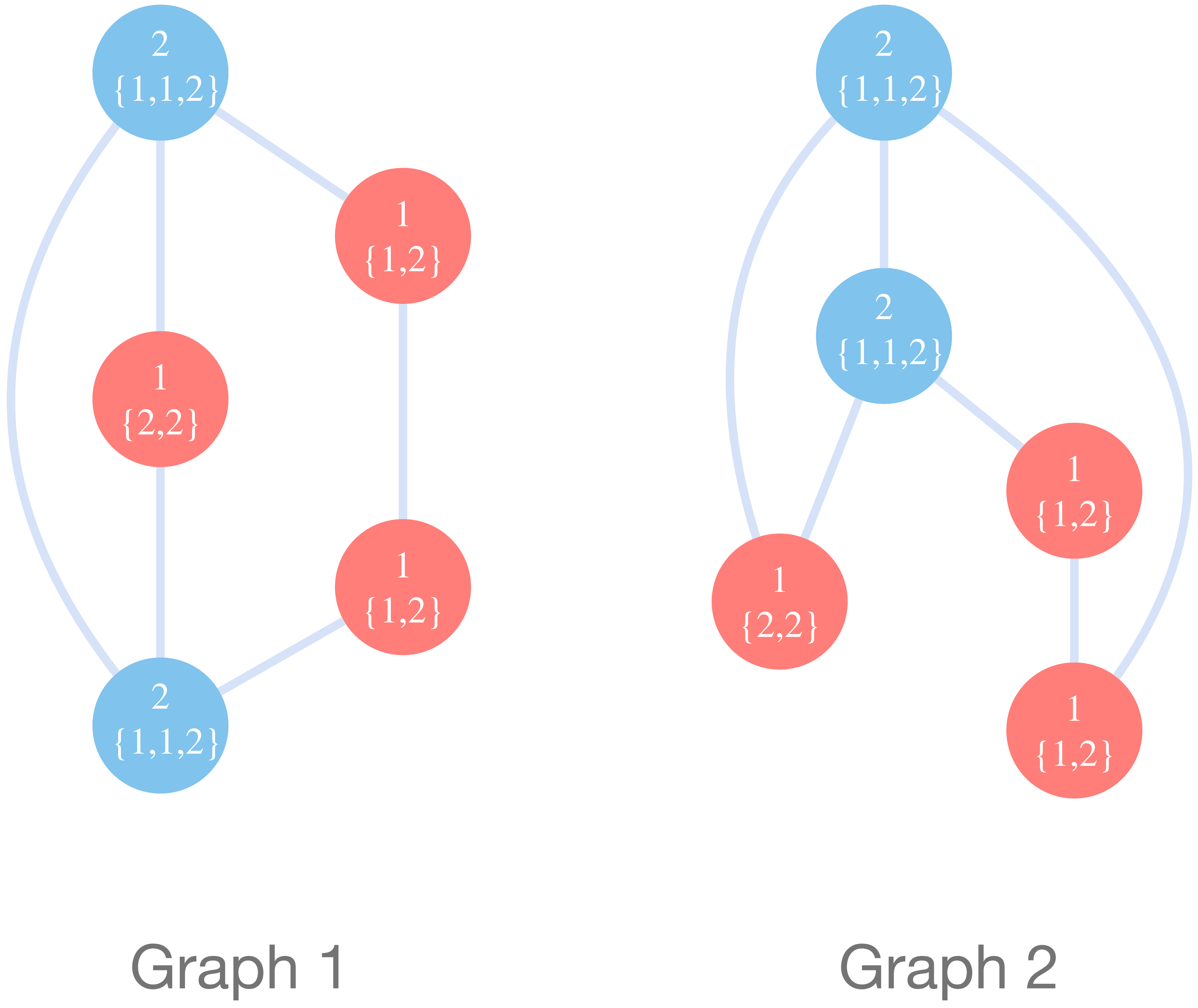


Figure 5: The Weisfeiler-Lehman Isomorphism Algorithm

WL Isomorphism Test

Algorithm[3]

- Initialization: Set node features x_v^0 to original graph labels

- For $t = 0,\ldots,n-1$, repeat

  - For each node v form a multi set S_v^t of the labels of all neighbors

$$S_v^t = \{x_u^t \,|\, u \in \mathcal{N}(v)\}$$

  - Map each pair of label x_v^t and multi set S_v^t to a new label x_v^{t+1} (e.g. via a hash function)

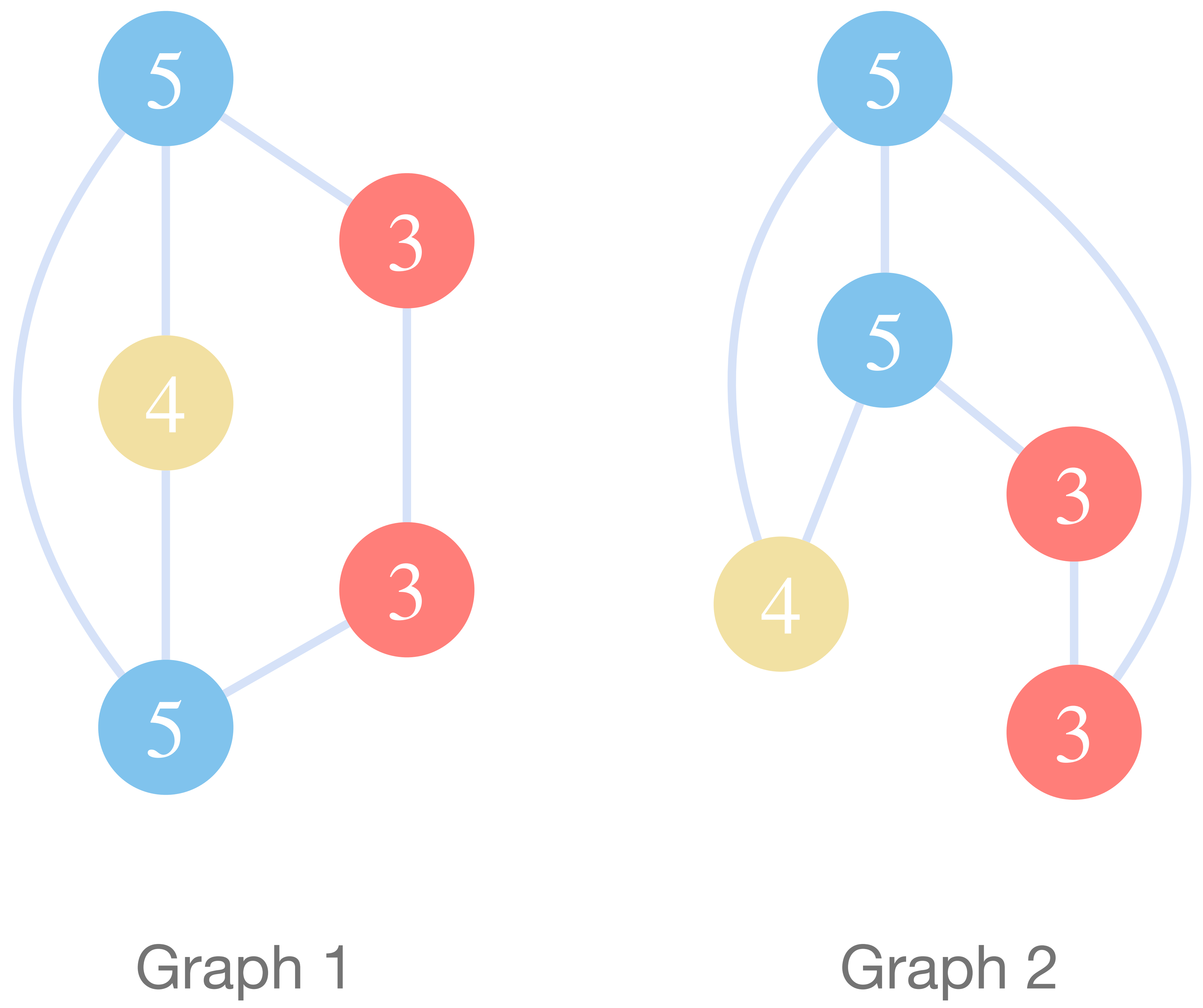


Figure 5: The Weisfeiler-Lehman Isomorphism Algorithm

WL Isomorphism Test

Algorithm[3]

- Initialization: Set node features x_v^0 to original graph labels
- For $t = 0, \ldots, n-1$, repeat
  - For each node v form a multi set S_v^t of the labels of all neighbors

$$S_v^t = \{x_u^t \mid u \in \mathcal{N}(v)\}$$

  - Map each pair of label x_v^t and multi set S_v^t to a new label x_v^{t+1} (e.g. via a hash function)

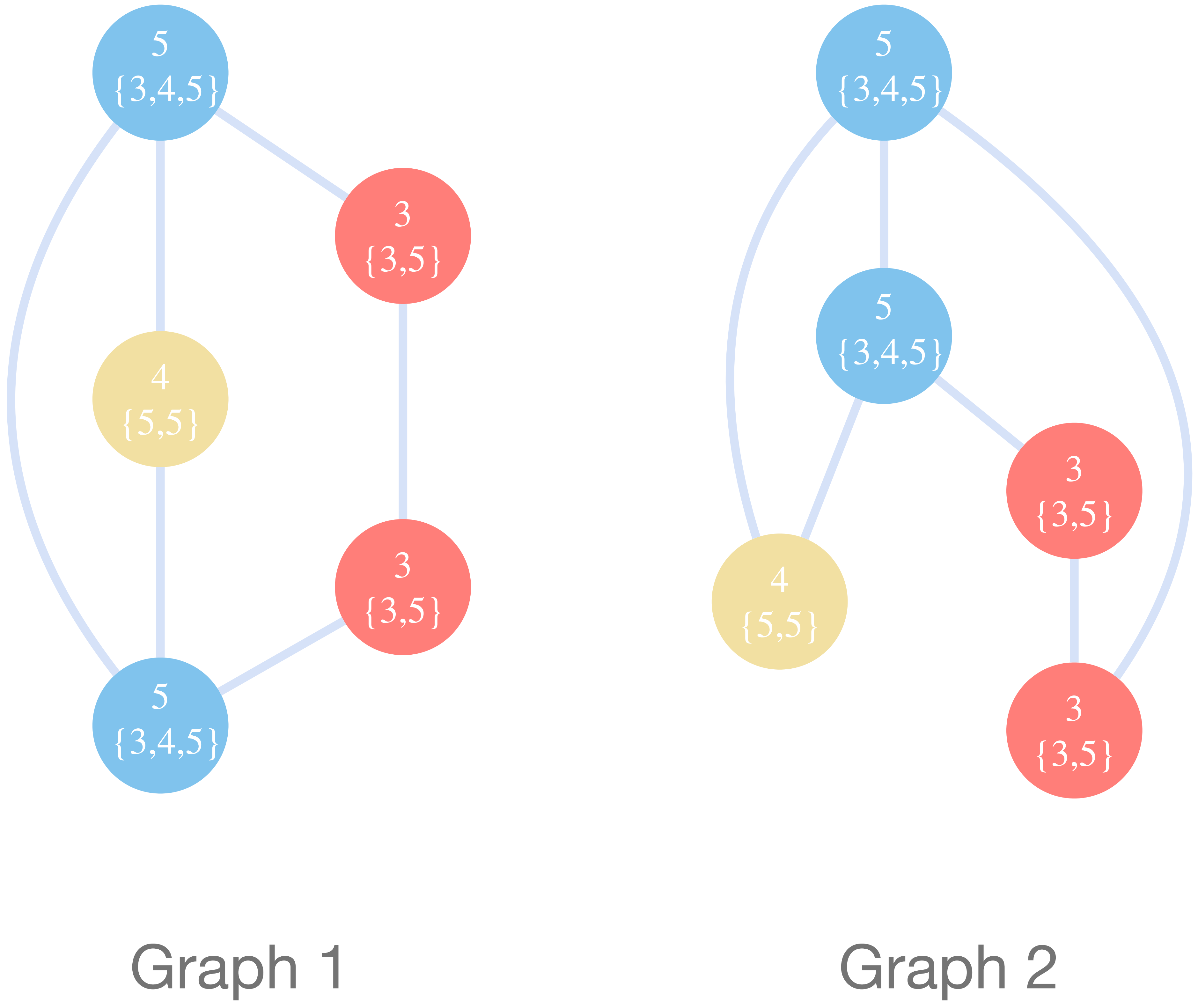


Figure 5: The Weisfeiler-Lehman Isomorphism Algorithm

WL Isomorphism Test

Algorithm[3]

- Initialization: Set node features x_v^0 to original graph labels
- For $t = 0, \ldots, n-1$, repeat
  - For each node v form a multi set S_v^t of the labels of all neighbors
  $$S_v^t = \{x_u^t \,|\, u \in \mathcal{N}(v)\}$$
  - Map each pair of label x_v^t and multi set S_v^t to a new label x_v^{t+1} (e.g. via a hash function)
  - Terminate if assignments of nodes to labels did not change from previous iteration

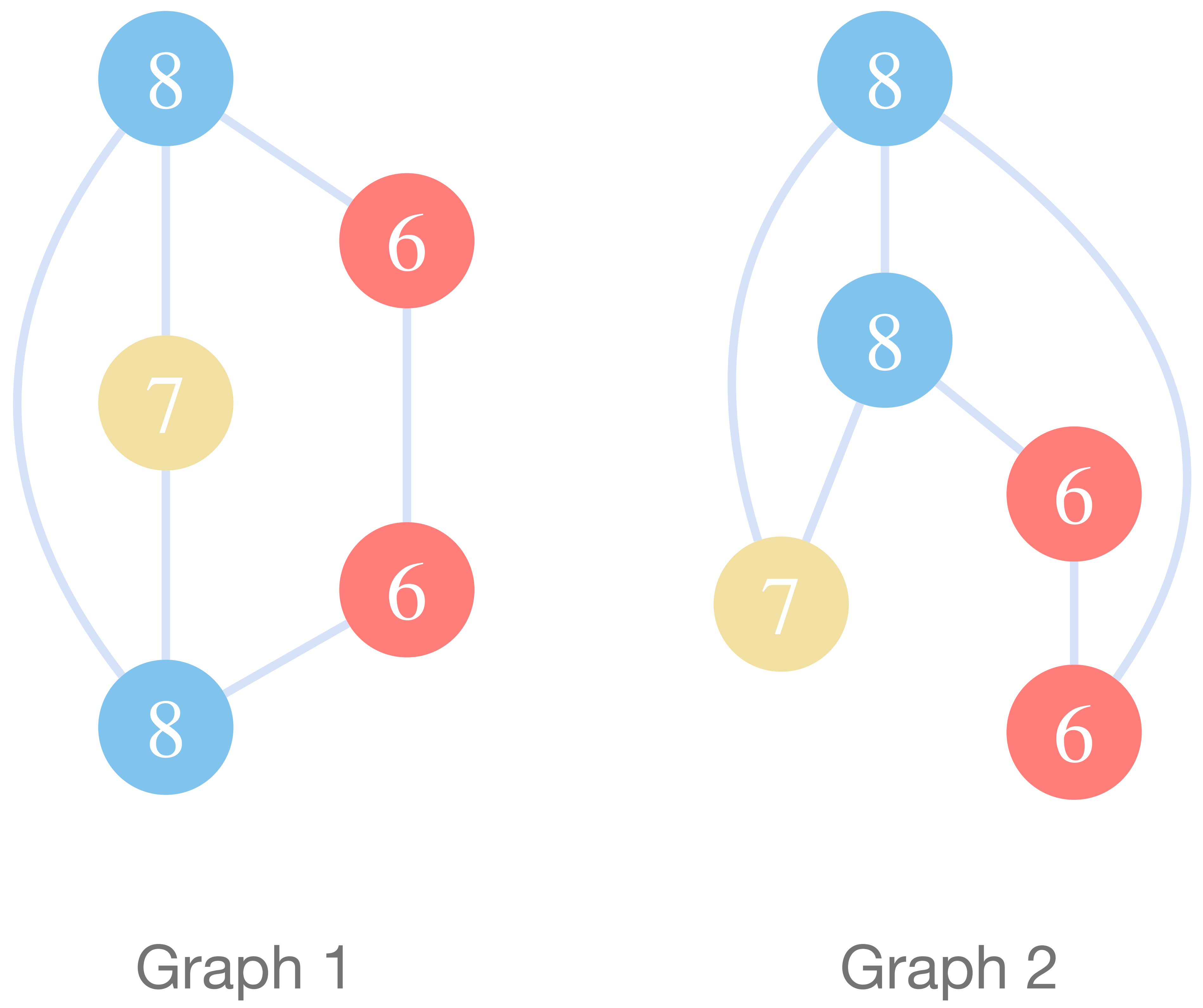
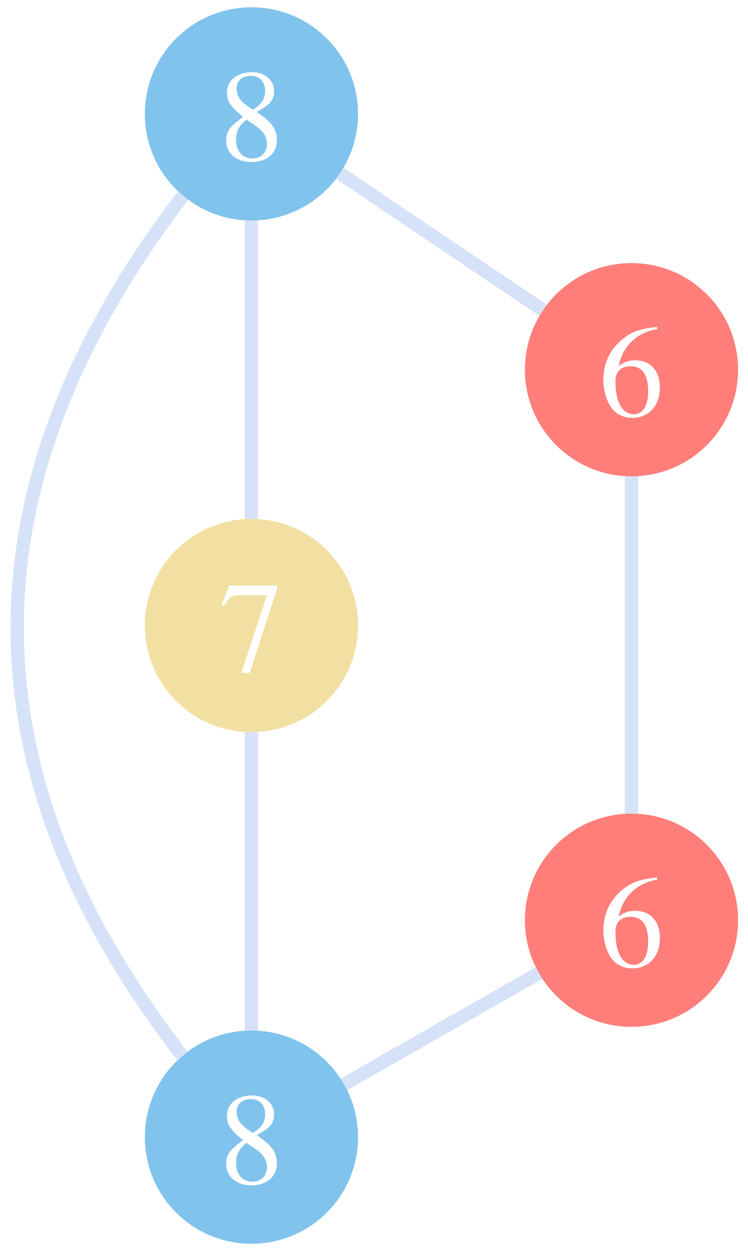


Figure 5: The Weisfeiler-Lehman Isomorphism Algorithm

WL Isomorphism Test

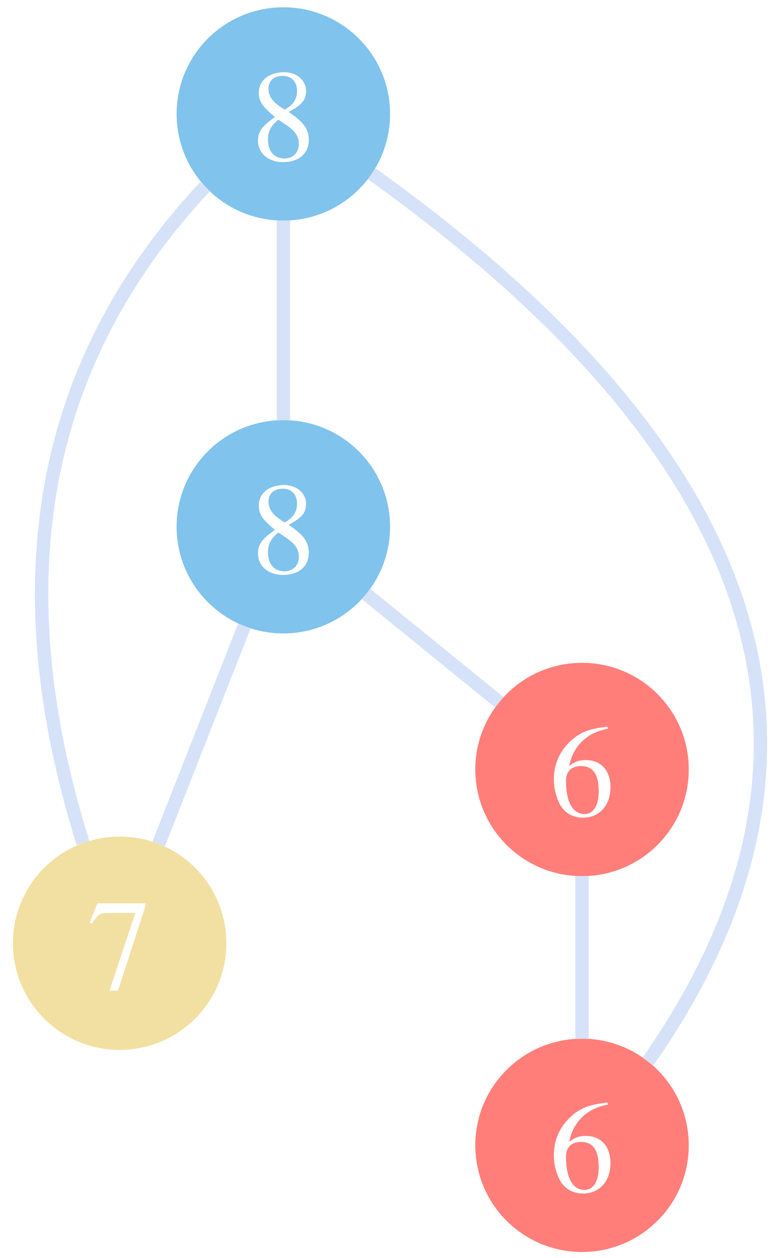
Algorithm[3]

- Initialization: Set node features x_v^0 to original graph labels
- For $t = 0, \ldots, n-1$, repeat
  - For each node v form a multi set S_v^t of the labels of all neighbors
  $$S_v^t = \{x_u^t \,|\, u \in \mathcal{N}(v)\}$$
  - Map each pair of label x_v^t and multi set S_v^t to a new label x_v^{t+1} (e.g. via a hash function)
  - Terminate if assignments of nodes to labels did not change from previous iteration



Graph 1

$\{6,6,7,8,8\}$



Graph 2

$\{6,6,7,8,8\}$

WL Isomorphism Test

Connection to GNNs

- Power of multi set function used in every layer of anonymous GNNs determines power in classifying graph isomorphism

Theorem 1

Every GNN is at most as powerful as the WL isomorphism test.[1]

Theorem 2

A GNN is as powerful as the WL isomorphism test if its layer aggregate, combine and readout functions are injective.[1]

GIN Model

Definition[1]

- GNN model that is as powerful as the WL isomorphism test

- Node update in each layer defined via

$$x_v^k = MLP^k\left(\left(1+\epsilon^k\right)x_v^{k-1} + \sum_{u\in\mathcal{N}(v)} x_u^{k-1}\right)$$

where ϵ^k are learnable parameters and MLP^k are learnable multi layer perceptrons

GIN Model

Results

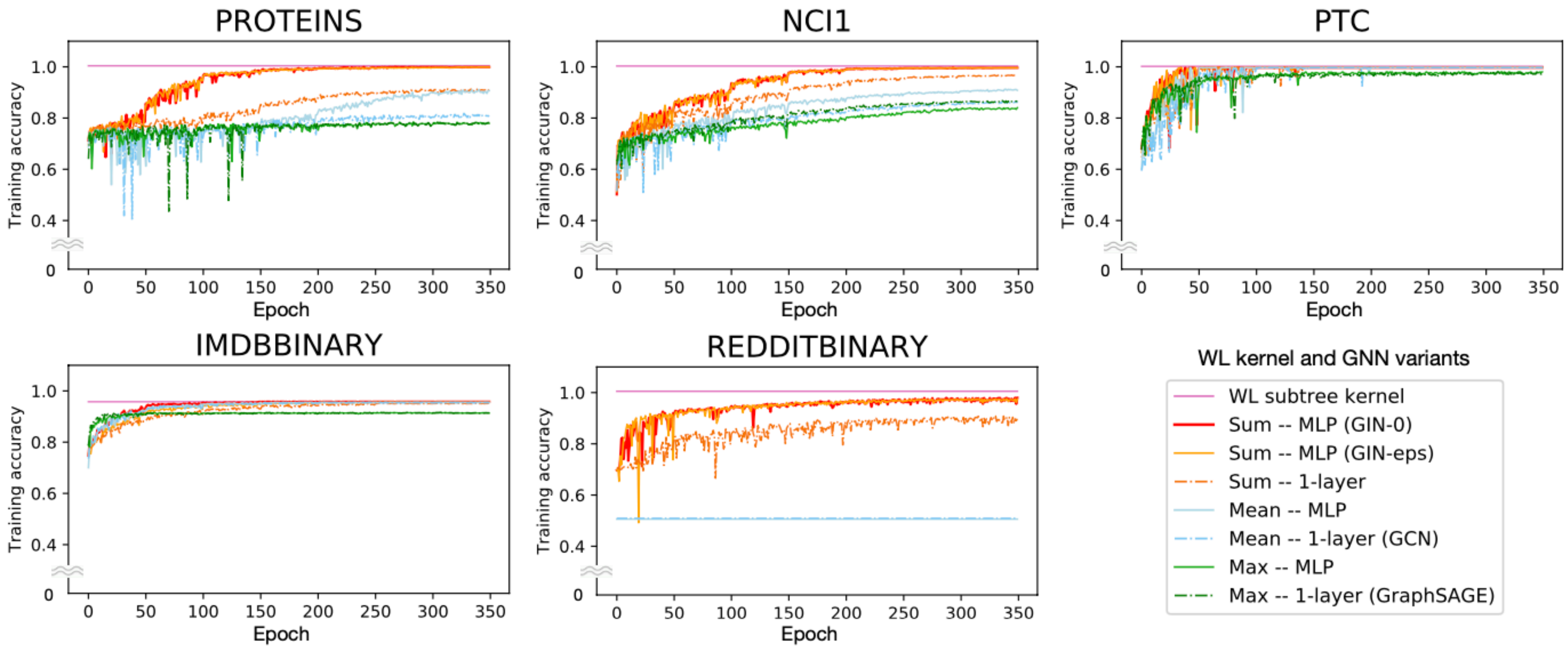


Figure 6: Performance of different GNN models on a selection of graph classification tasks[1]

Less Powerful Models

One-layer perceptron[1]

Lemma

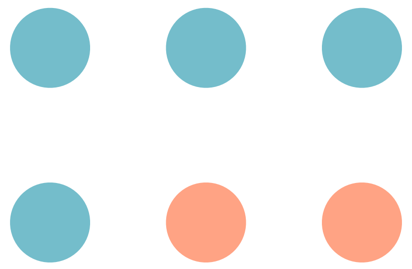
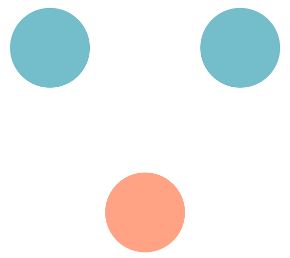
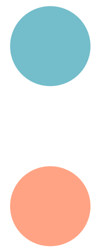
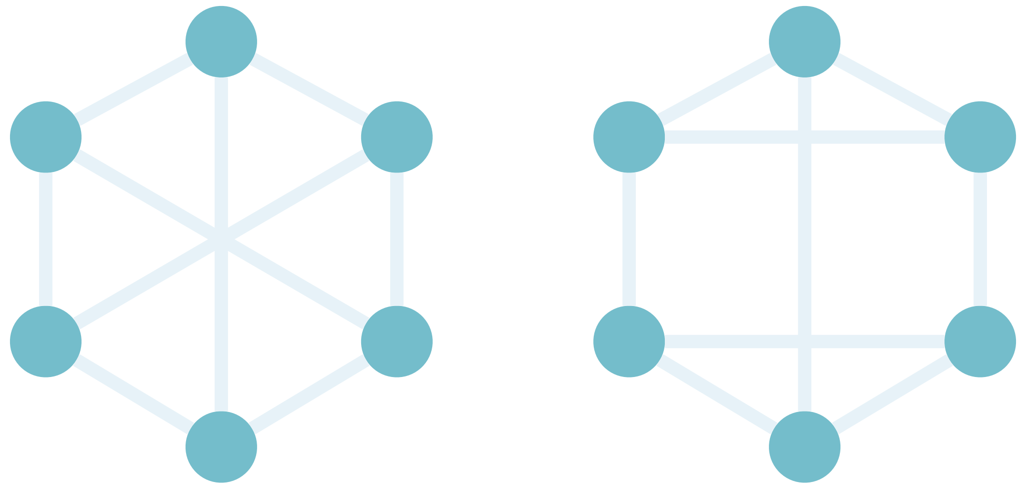
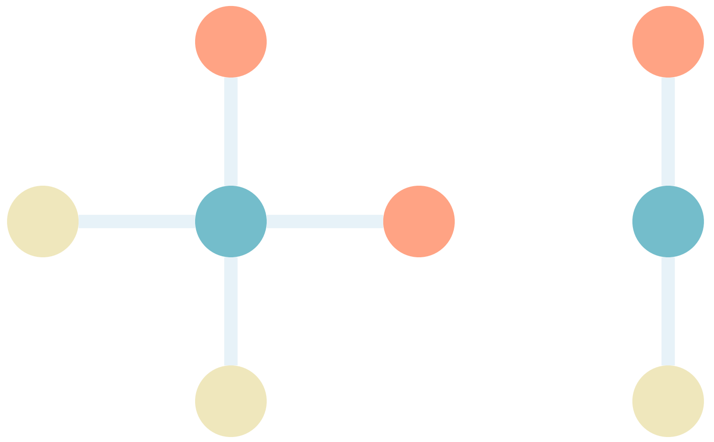
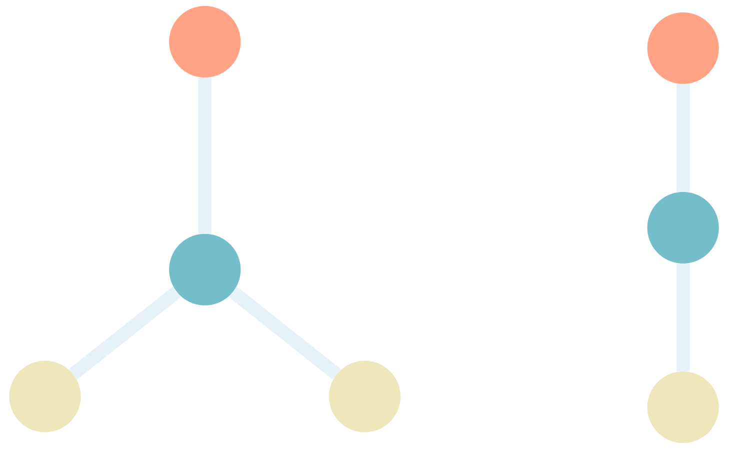
There exist finite multi sets $X_1 \neq X_2$, such that for any linear mapping W

$$\sum_{x \in X_1} ReLU(Wx) = \sum_{x \in X_2} ReLU(Wx)$$

- Linear model (without bias term) fails to distinguish between some multi sets
- One-layer perceptron is not a universal approximator of multi set functions (unlike MLP)

Less Powerful Models

Different Aggregation Schemes[1]

Aggregation Function	Sum	Mean	Max
Classification level	Multiset	Distribution	Set
Sample Input 			
Failure Example			

GNNs with port numbering

Non-anonymous GNNs

- Anonymous GNNs cannot distinguish between messages from different neighbors and are at most as powerful as the WL-test

- Idea: Assign port numbering to distinguish between different neighbours

Port Numbering

Definition: Port

A port of a graph G is a pair (v, i) where $v \in V$ and $i \in \{1,2,\ldots,\deg(i)\}$. We denote the set of all ports of G with $P(G)$.

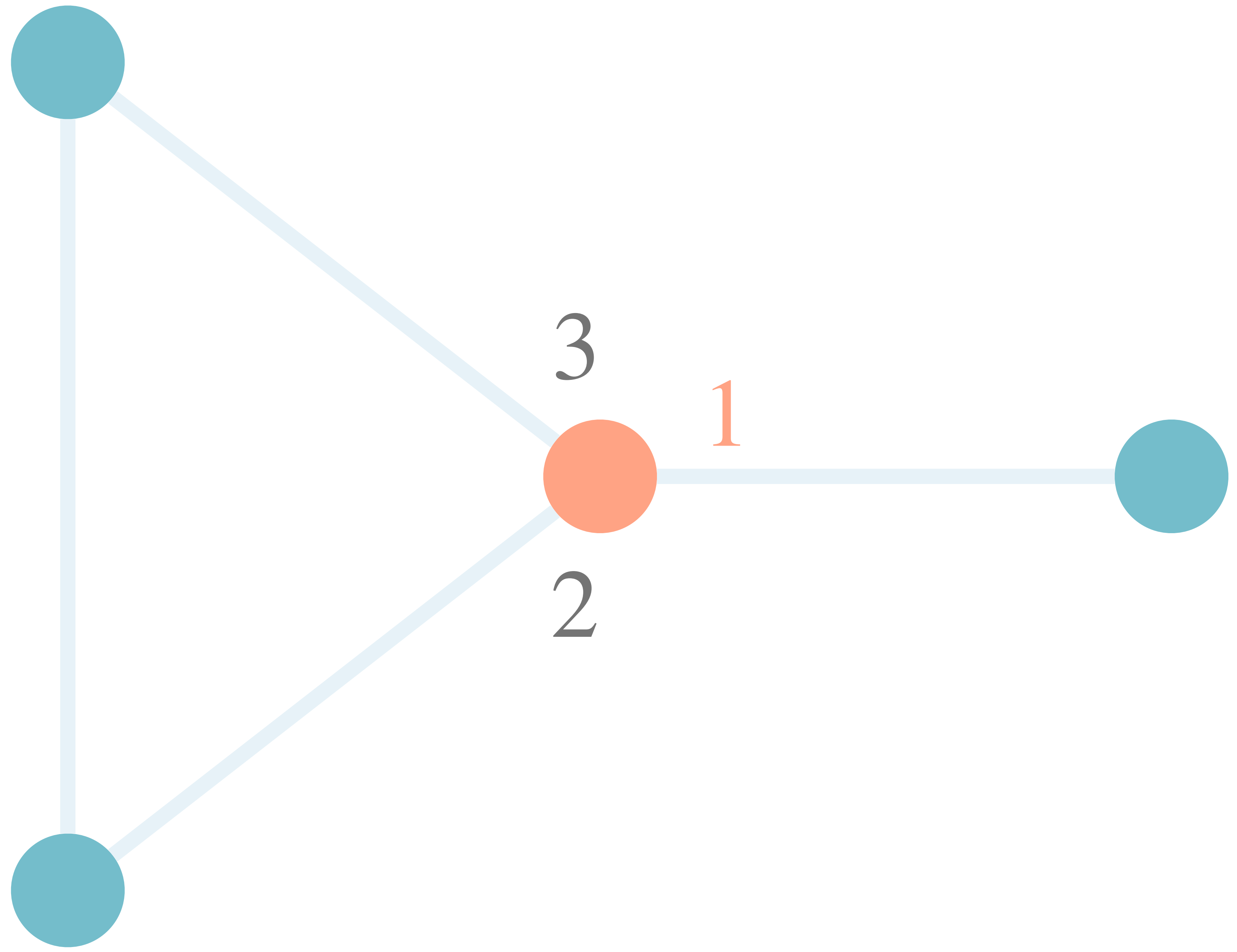


Figure 7: Example of a consistent port numbering[4]

Port Numbering

Definition: Port

A port of a graph G is a pair (v, i) where $v \in V$ and $i \in \{1,2,\ldots,\deg(i)\}$. We denote the set of all ports of G with $P(G)$.

Definition: Port Numbering

A port numbering is a function $p : P(G) \rightarrow P(G)$, such that for any edge (u, v), there exist i, j with $p(u, i) = (v, j)$.

We call p consistent if it is self-inverse, i.e. $p(p(v, i)) = (v, i)$.

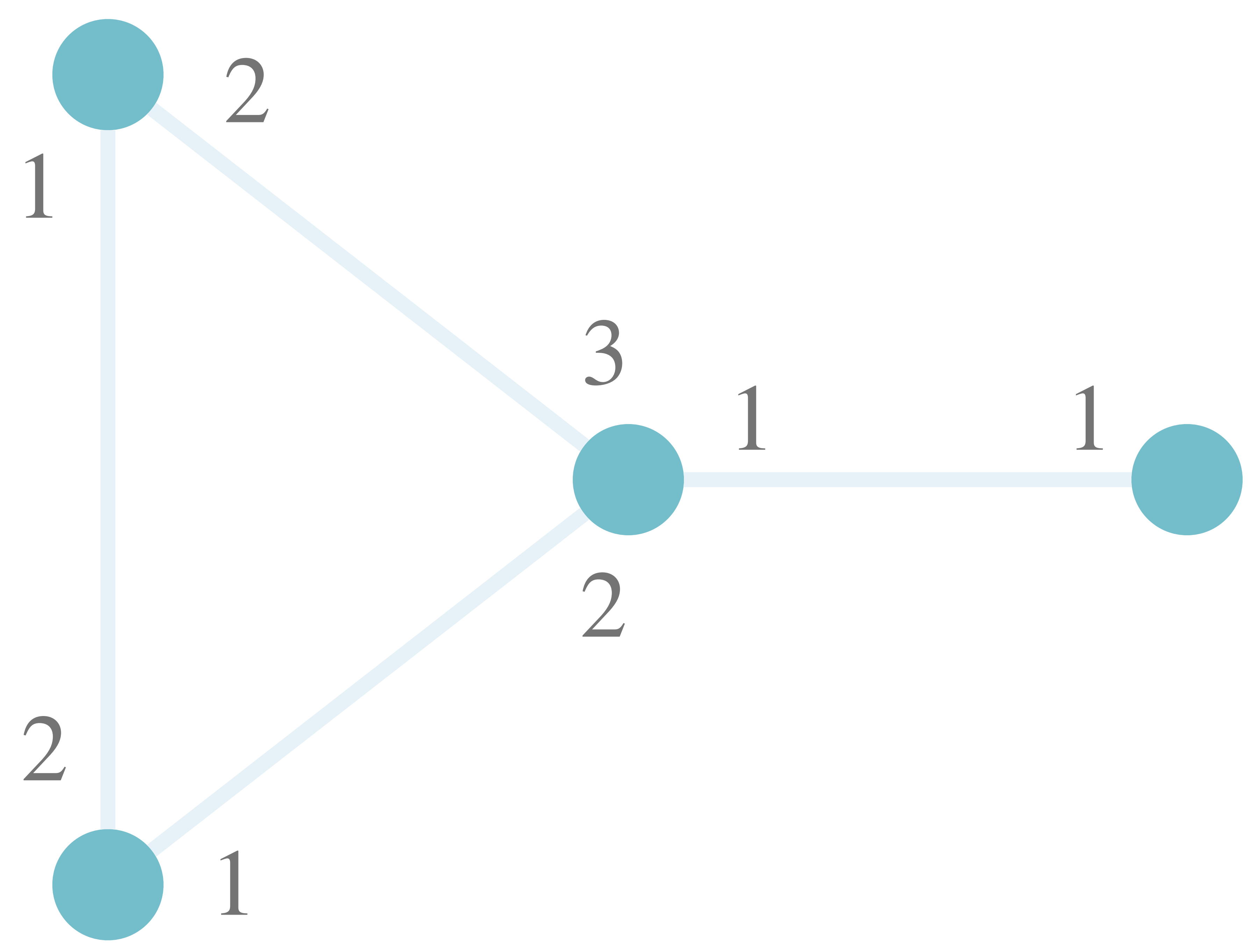


Figure 7: Example of a consistent port numbering[4]

Vector-vector consistent GNNs

- Let p be a consistent port numbering and denote its two components by p_1, p_2, i.e.

$$p(v,i) = \big(p_1(v,i), p_2(v,i)\big)$$

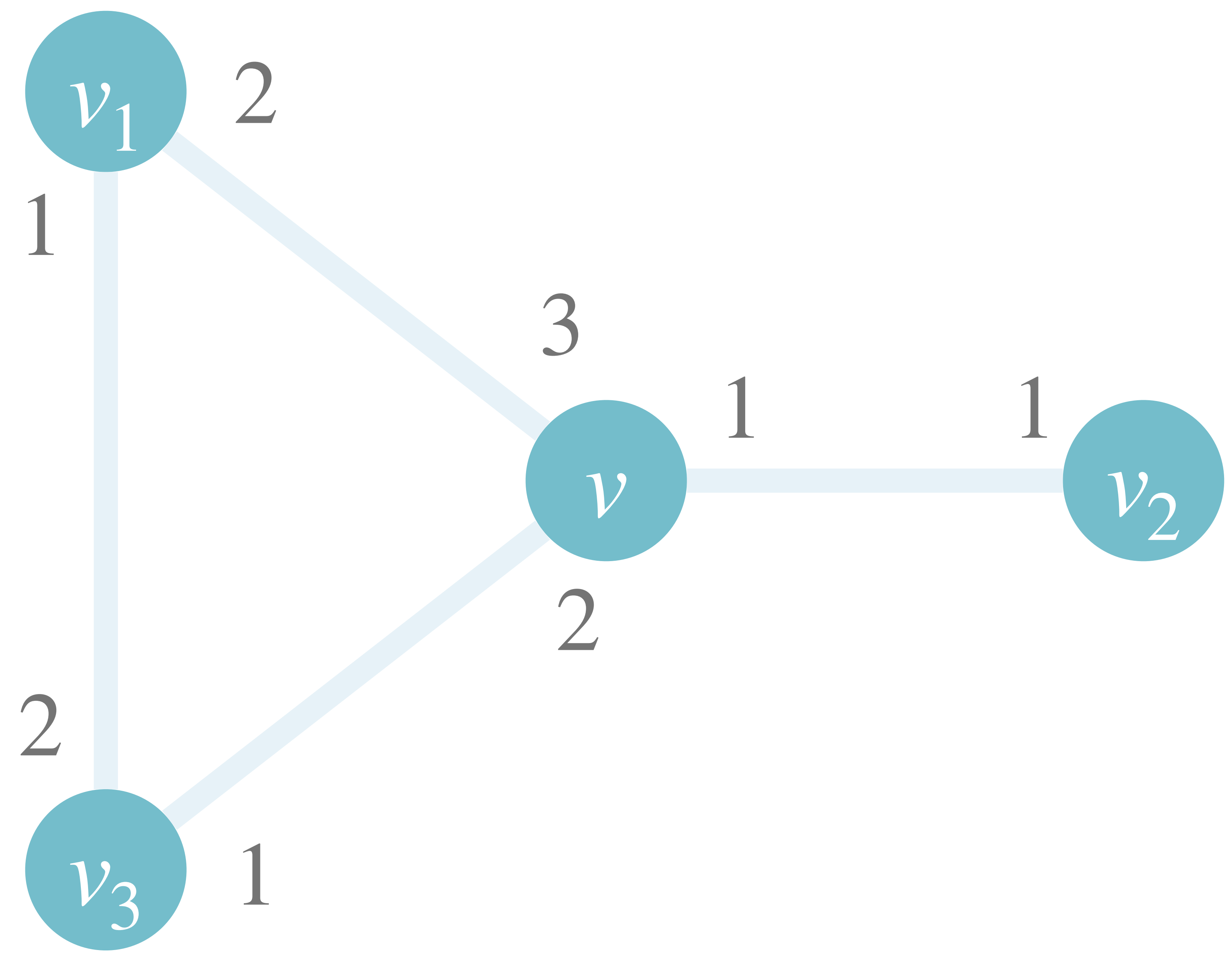
- Extend anonymous GNNs by including consistent port numbering in layer input

$$x_v^{t+1} = f_\theta\left(x_v^t, \left(x_{p_1(v,1)}^t, p_2(v,1)\right), \left(x_{p_1(v,1)}^t, p_2(v,1)\right), \ldots \left(x_{p_1(v,\Delta)}^t, p_2(v,\Delta)\right)\right)$$

- Port numbering can be computed beforehand in linear time

Vector-vector consistent GNNs

Example



$$p(v,1) = (v_2,1)$$

$$p(v,2) = (v_3,1)$$

$$p(v,3) = (v_1,2)$$

$$x_v^{t+1} = f_\theta\left(x_v^t, \left(x^t_{p_1(v,1)}, p_2(v,1)\right), \left(x^t_{p_1(v,1)}, p_2(v,1)\right), \ldots \left(x^t_{p_1(v,\Delta)}, p_2(v,\Delta)\right)\right)$$

$$= f_\theta\left(x_v^t, \left(x^t_{v_2}, 1\right), \left(x^t_{v_3}, 1\right), \left(x^t_{v_1}, 2\right)\right)$$

Vector-vector consistent GNNs

CPNGNNs

- Authors of [3] introduce Consistent Port Numbering Graph Neural Networks (CPNGNNS)

$$x_v^{t+1} = ReLU\left(W^t \cdot CONCAT\left(x_v^t, x_{p_1(v,1)}^t, p_2(v,1), x_{p_1(v,1)}^t, p_2(v,1), \ldots x_{p_1(v,\Delta)}^t, p_2(v,\Delta)\right)\right)$$

$$x_v^{final} = MLP\left(x_v^T\right)$$ (in final layer)

- CPNGNNs (and VVC-GNNs) are strictly more powerful than regular GNNs

- Example: Finding single leaf problem

Vector-vector consistent GNNs

CPNGNNs

- Authors of [3] introduce Consistent Port Numbering Graph Neural Networks (CPNGNNS)

$$x_v^{t+1} = ReLU\left(W^t \cdot CONCAT\left(x_v^t, x^t_{p_1(v,1)}, p_2(v,1), x^t_{p_1(v,1)}, p_2(v,1), \ldots x^t_{p_1(v,\Delta)}, p_2(v,\Delta)\right)\right)$$

$$x_v^{final} = MLP\left(x_v^T\right)$$ (in final layer)

- CPNGNNs (and VVC-GNNs) are strictly more powerful than regular GNNs

- Example: Finding single leaf problem

Vector-vector consistent GNNs

Single Leaf Problem[4]

- Input: star graph
- Output: A single marked leaf node
- Basic GNNs fail since different leaf nodes cannot be distinguished and output coincides

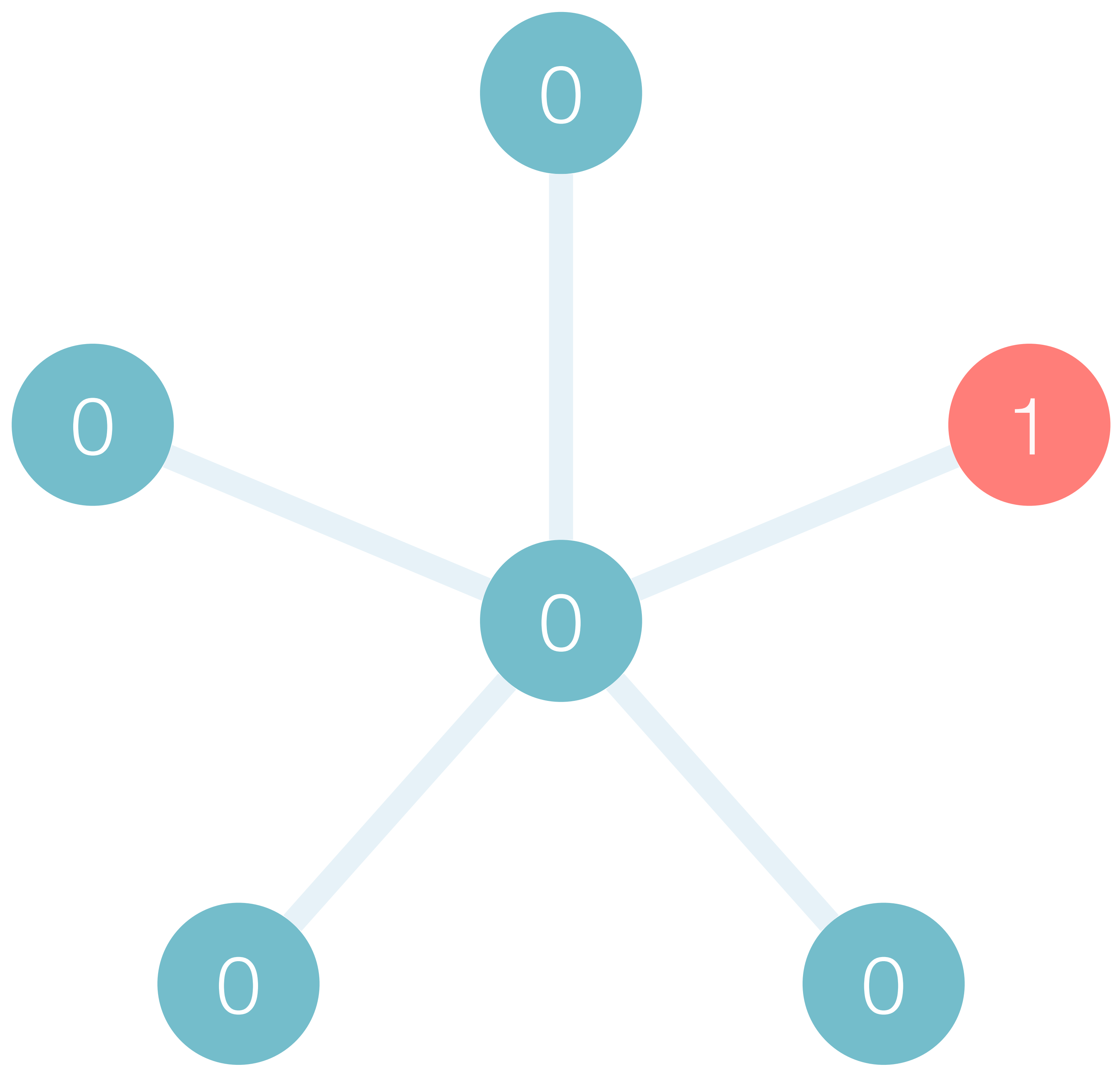


Figure 8: Example instance of single leaf problem

Distributed Computing

GNNs with unique vertex IDs

- Strictly more powerful than other GNN classes
- Turing universal under certain conditions
- Problems arise during training since GNNs with unique vertex IDs do not generalise well
- Limitations for GNNs with unique vertex IDs also hold for other types of GNNs

LOCAL and CONGEST

- Distributed computing models with unique node IDs

1. Communication network
   - Represented by graph G
   - Each node represents a machine and communicates only with its neighbors

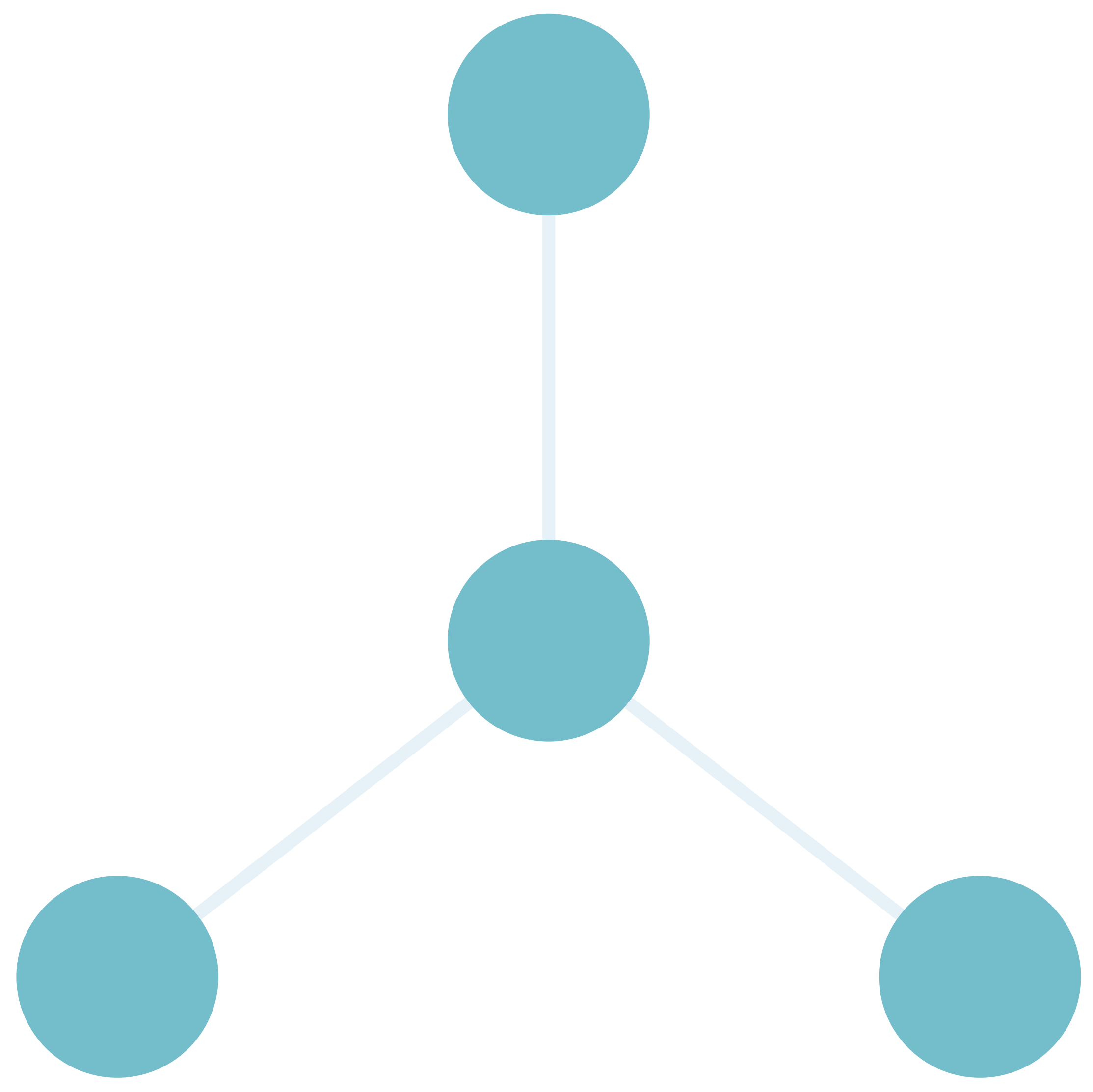


Figure 9: The LOCAL model of computation

LOCAL and CONGEST

- Distributed computing models with unique node IDs

1. Communication network
   - Represented by graph G
   - Each node represents a machine and communicates only with its neighbors

2. Synchronous computation
   - Computation performed in synchronous rounds where each round consists of two steps

     i) Propagate messages between neighbors
     ii) Perform arbitrarily powerful computation for each node

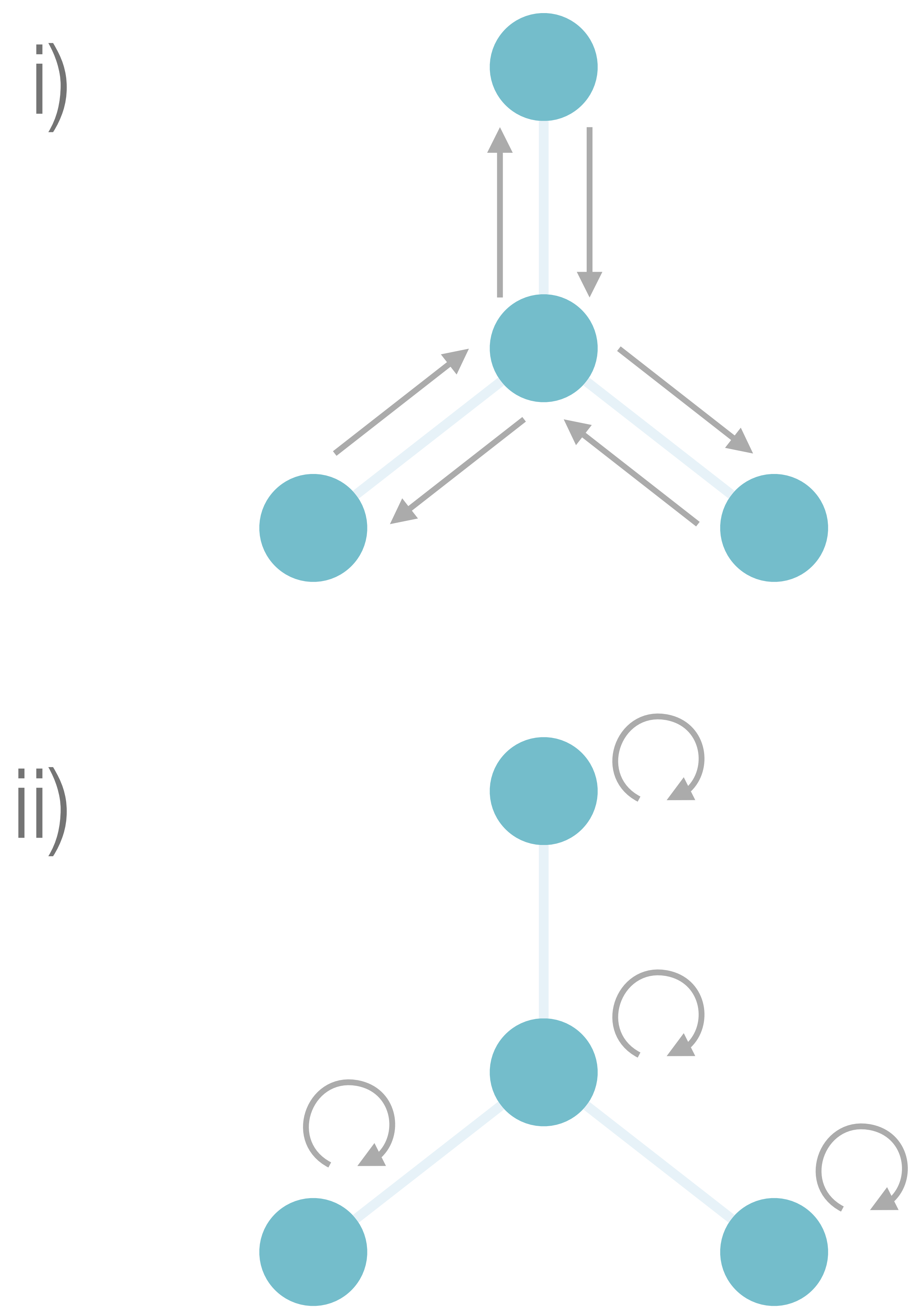


Figure 9: The LOCAL model of computation

LOCAL and CONGEST

- Distributed computing models with unique node IDs

1. Communication network
   - Represented by graph G
   - Each node represents a machine and communicates only with its neighbors
2. Synchronous computation
   - Computation performed in synchronous rounds where each round consists of two steps

     i) Propagate messages between neighbors
     ii) Perform arbitrarily powerful computation for each node
3. Message size
   - In CONGEST model: restricted in size to b bits

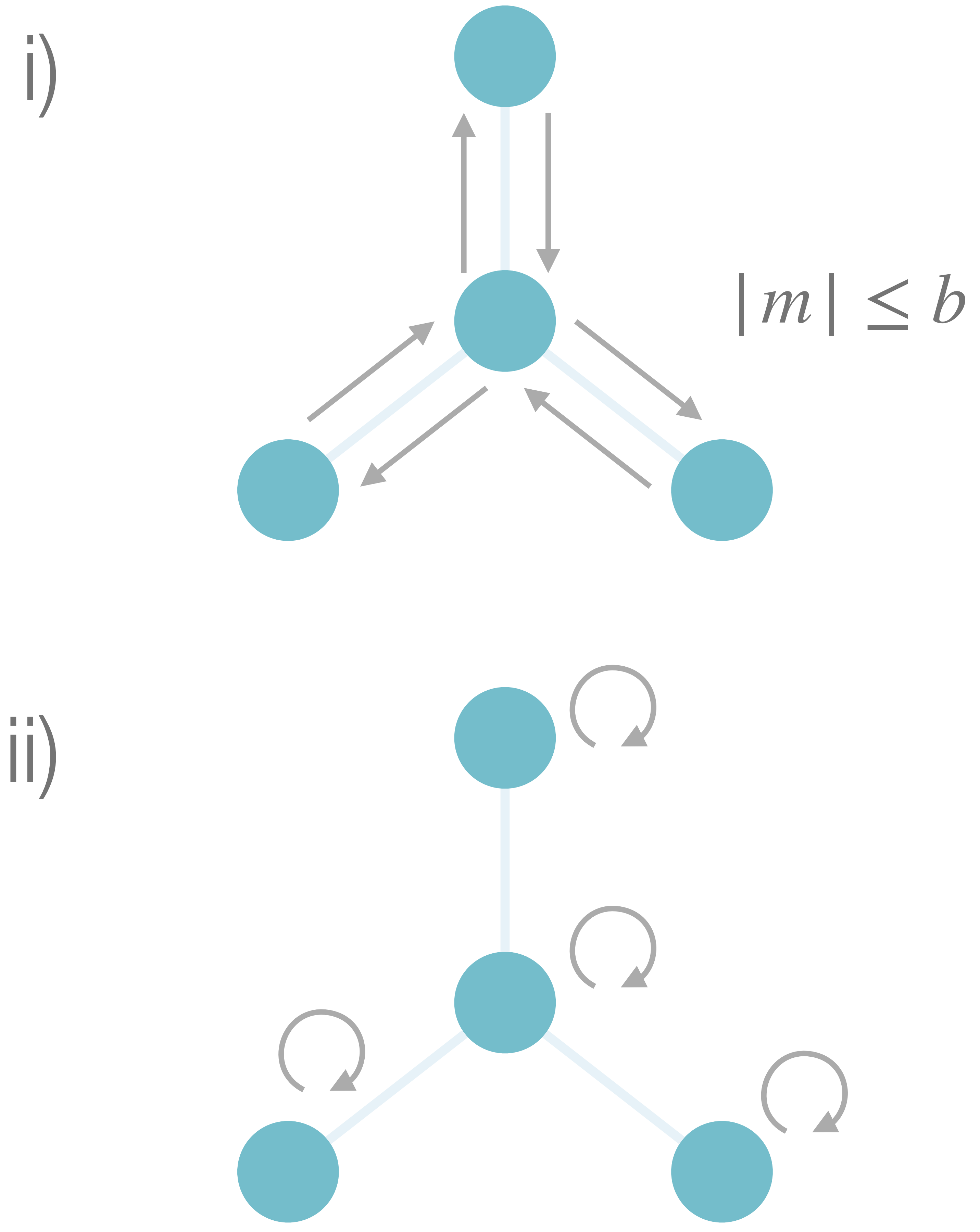


Figure 9: The LOCAL model of computation

LOCAL and CONGEST

Connection to GNNs

Theorem

Message passing GNNs with unique vertex IDs and Turing complete aggregate and combine functions are equivalent to algorithms in the LOCAL model of computation.[5]

- Allows us to infer limits for the computational complexity of GNNs by leveraging results from the LOCAL model
- Similarly, we can infer limits for GNNs with limited width, using results from the CONGEST model

Requirements for Turing Universality

Message passing GNNs are Turing universal under the following conditions

i) Each node is uniquely identified

ii) The aggregate and combine functions are Turing complete

iii) The depth of the GNN is larger than the diameter of the input graph

iv) The width of the GNN is unbounded

- Note that universality in the case of graph level classification is trivial if the $READOUT$ function is Turing complete

Requirements for Turing Universality

Message passing GNNs are Turing universal under the following conditions

i) Each node is uniquely identified

ii) The aggregate and combine functions are Turing complete

Required for equivalence to LOCAL model

iii) The depth of the GNN is larger than the diameter of the input graph

iv) The width of the GNN is unbounded

Required for Turing universality in LOCAL model

- Note that universality in the case of graph level classification is trivial if the $READOUT$ function is Turing complete

Limits from CONGEST model

Theorem

If a problem P cannot be solved in less than d rounds in $CONGEST$ using messages of at most b bits, then P cannot be solved by a GNN of depth d and width $w = \mathcal{O}(b/\log(n))$.[5]

- Yields limits for the depth and width of a GNN, even for local problems
- Example: k-cycle classification for $k \geq 4$ requires depth

$$d = \Omega\left(\sqrt{n}/(w \log n)\right) \qquad \text{if } k \text{ even,}$$

$$d = \Omega\left(n/(w \log n)\right) \qquad \text{if } k \text{ odd.}$$

Limits from CONGEST model

Experimental results

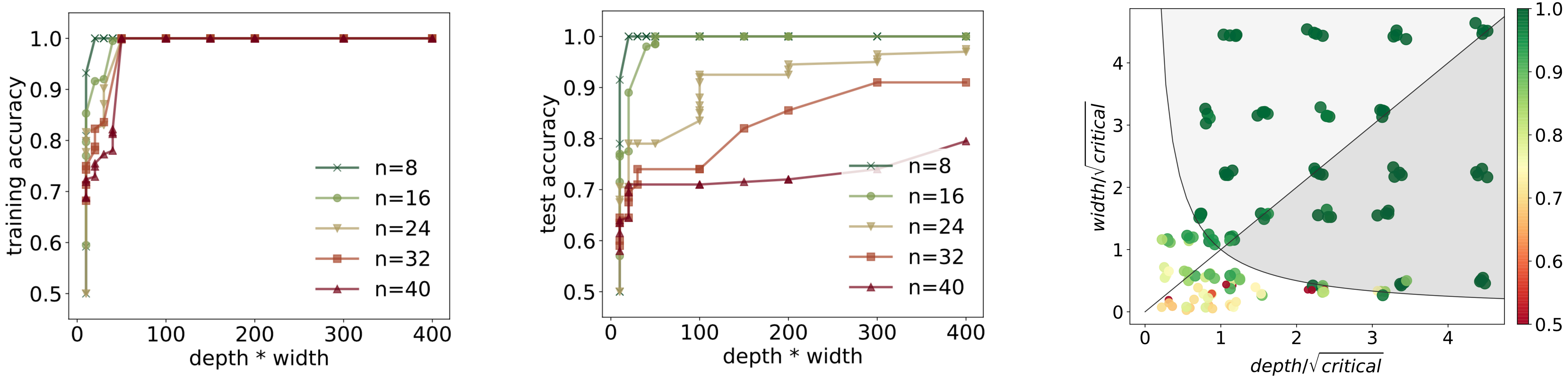


Figure 10: Performance of GNNs with different depth and width on the 4-cycle problem (determining whether a graph contains a 4-cycle).[5]

Communication Capacity and Limitations

Communication capacity

- Computational power of GNN dependent on its depth and width: motivates generalising notion of communication complexity

- Assume that each node feature vector takes values in some finite alphabet $\mathcal{S}$ with $s = |\mathcal{S}|$ symbols

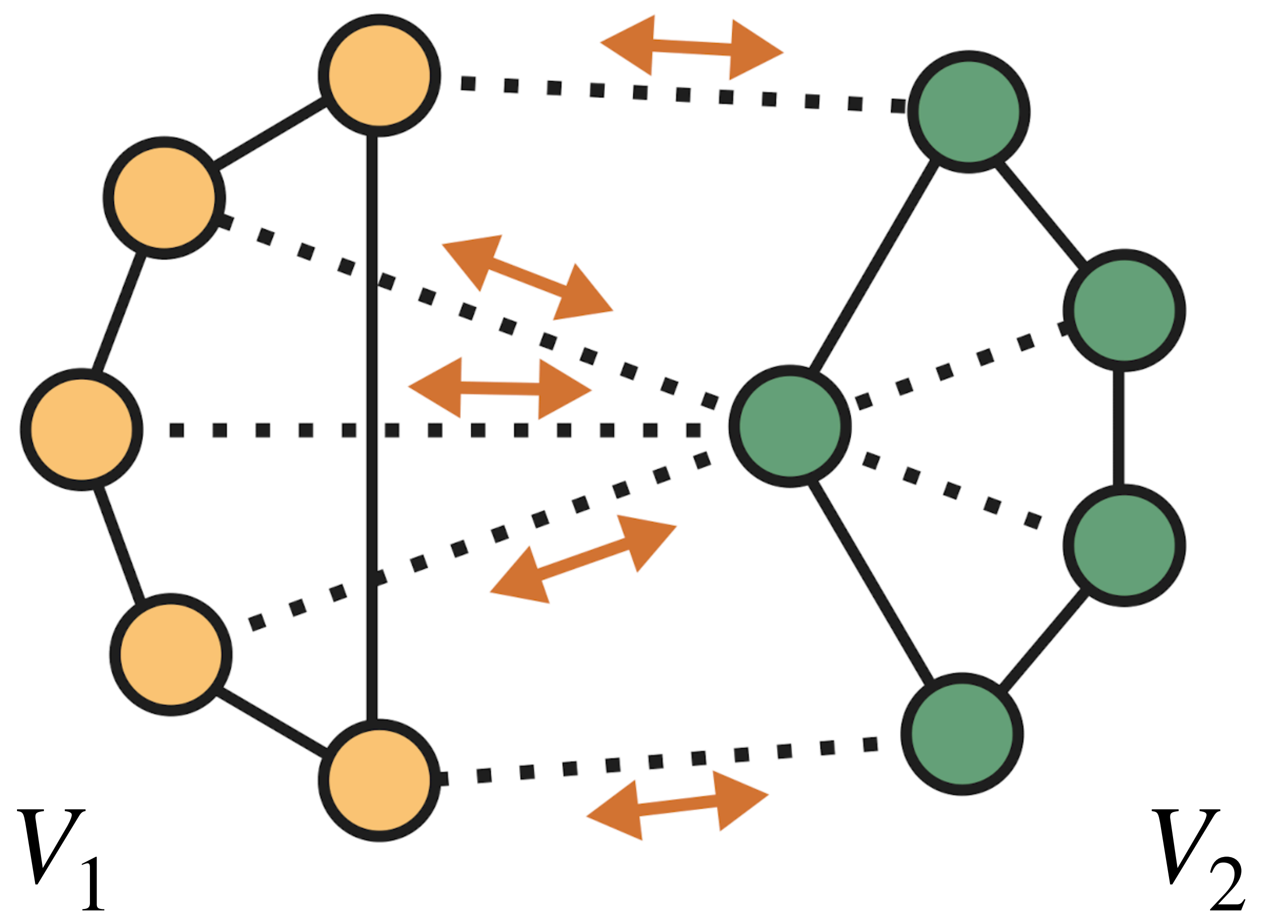


Figure 11: Example of a graph partition

Definition

Let g be a GNN and fix a graph $G = (V, E)$. For any two disjoint sets $V_1, V_2 \subseteq V$, the communication capacity c_g of g (with respect to G, V_1, V_2) is the maximum number of symbols that can be transmitted from V_1 to V_2 and vice versa.[6]

Communication capacity

- The communication capacity of a GNN with respect to the partition V_1, V_2 depends on

   i) Its width w and its depth d

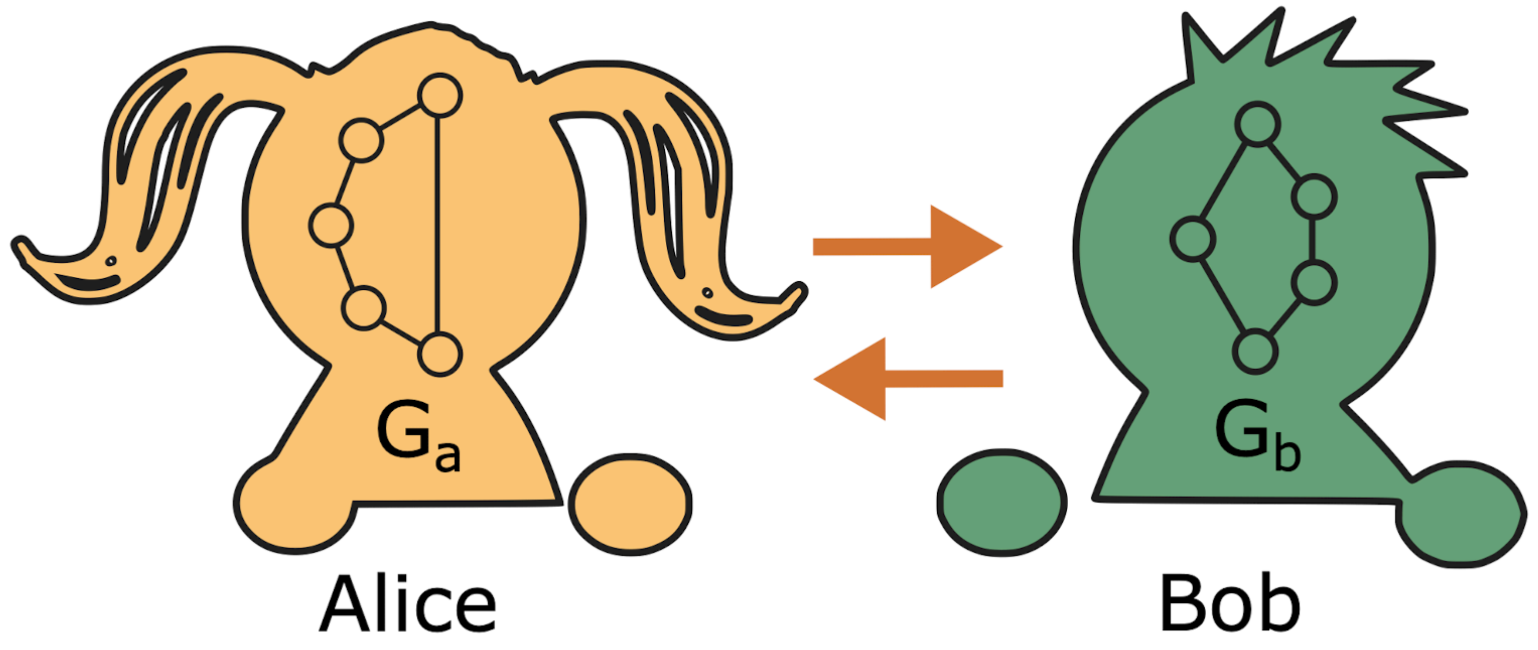
   ii) The size of messages passed in each layer

   iii) The size of its global state (if included)

   iv) The smallest cut separating the two subsets V_1, V_2

Communication complexity

- Two players with respective inputs x_a, x_b attempt to compute a function $f(x_a, x_b)$
- A communication protocol π determines the sequence of exchanged symbols between the players
- The number of exchanged symbols is denoted by $|\pi(x_a, x_b)|$



$x_a \in X_a$ $\qquad$ $x_b \in X_b$

Figure 12: Two players with respective inputs x_a, x_b, (here depicted as graphs G_a, G_b)

Definition

The communication complexity c_f of f corresponds to the minimum worst-case length of any protocol that computes f [6]

$$c_f = \min_{\pi} \max_{(x_a, x_b) \in X_a \times X_b} |\pi(x_a, x_b)|$$

Hardness of Graph Isomorphism Problem

- We can relate communication capacity and complexity to derive limitations of GNNs for graph isomorphism
- Idea: Consider two random graphs connected by a small amount of edges
- Results also hold in expectation for these specific sets of graphs

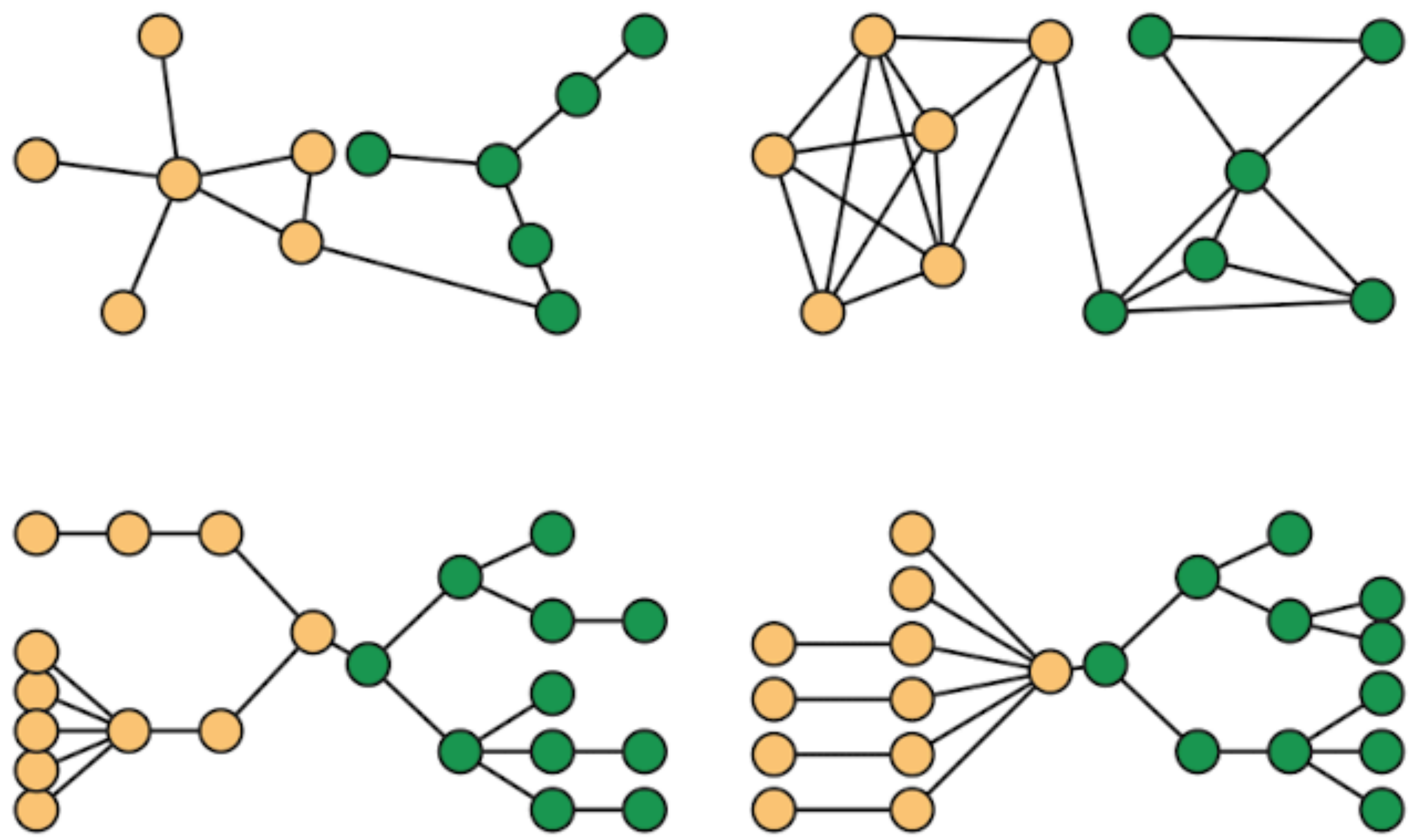


Figure 13: Sample graphs for the hardness proof of graph isomorphism for GNNs

Theorem

Let g be a GNN using a majority-voting or consensus based $READOUT$ function. To compute the isomorphism class of every graph/tree of n nodes, it must be that $c_g = \Omega(n^2)/c_g = \Omega(n)$.[6]

Hardness of Graph Isomorphism Problem

Empirical Results

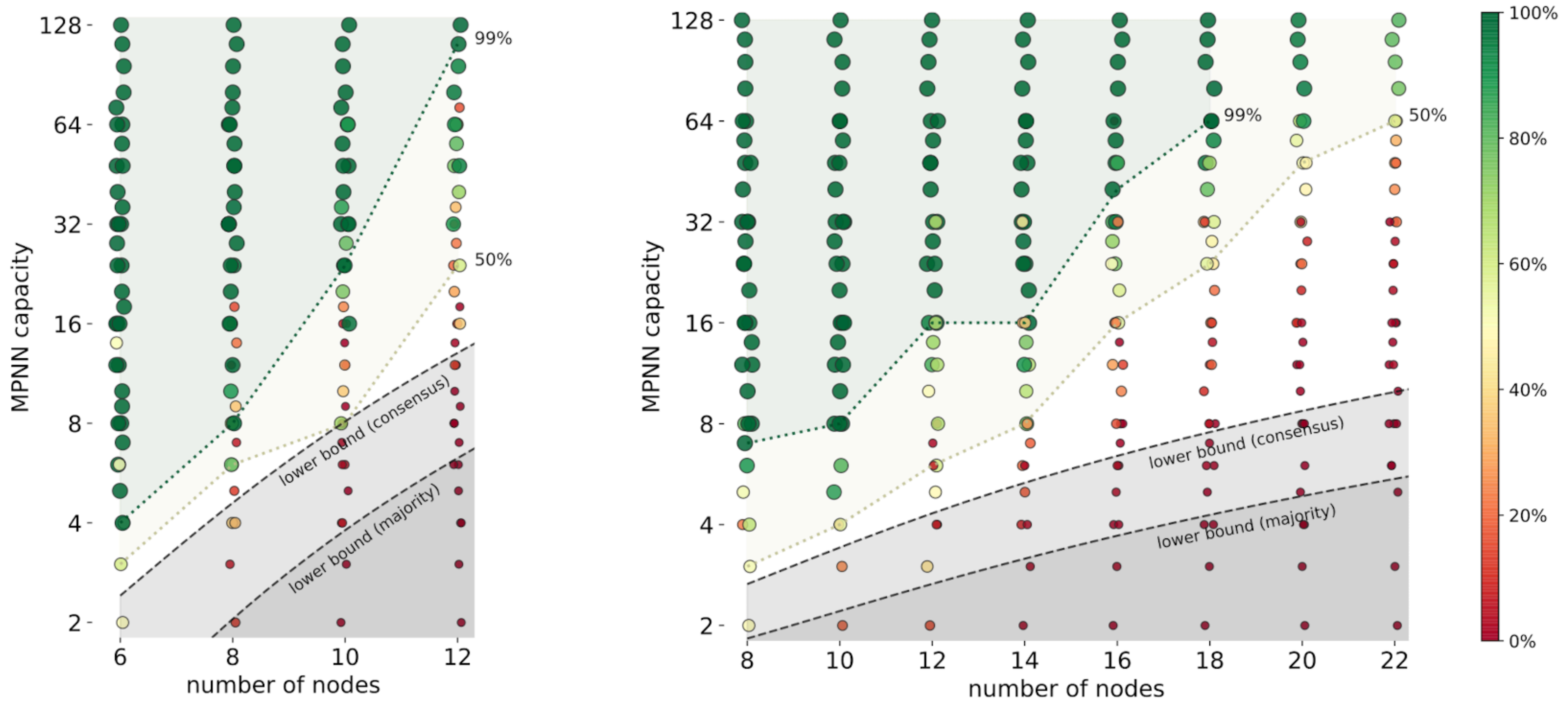


Figure 14: Performance of GNNs with different communication capacity on the graph isomorphism problem for a sample set of general graphs (a) and a set of trees (b)[6].

Oversquashing

Oversquashing

Definition

The problem radius r of a graph problem corresponds to its required range of interaction.

- GNN requires at least $K \geq r$ layers
- Size of receptive field of a node grows exponentially in the number of layers K

$$|\mathcal{N}_v^K| = \mathcal{O}(\exp(K))$$

- For fixed length feature vector x_v^t this leads to an exponential bottleneck

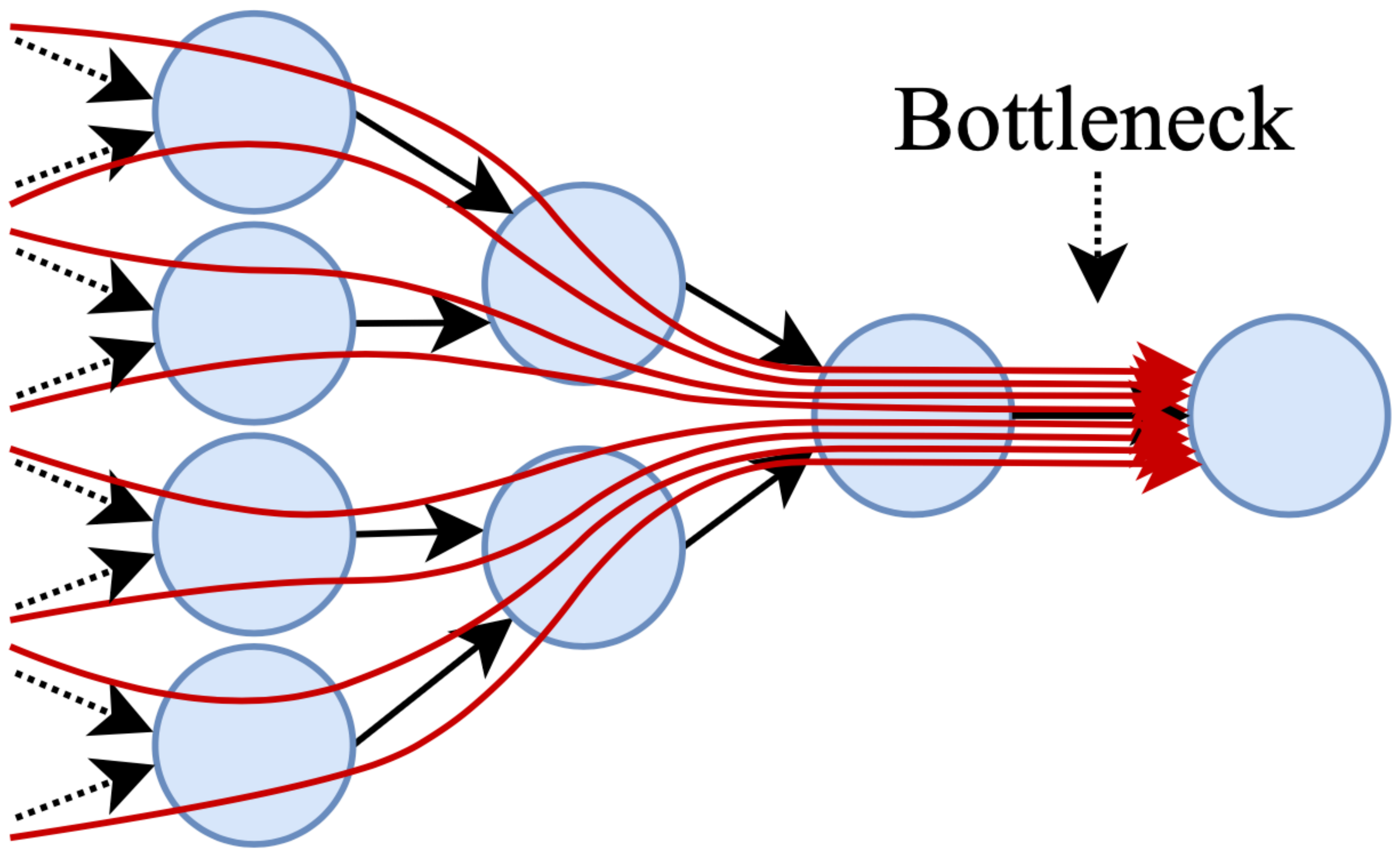


Figure 15: The bottleneck in GNNs with many layers[7]

Oversquashing

Example Problem

- In the *NeighborsMatch* problem the goal is to predict the label of a node based on its degree
- Solution requires propagation of information from all labeled nodes to target node
- Leads to bottleneck that prevents fitting the training data perfectly

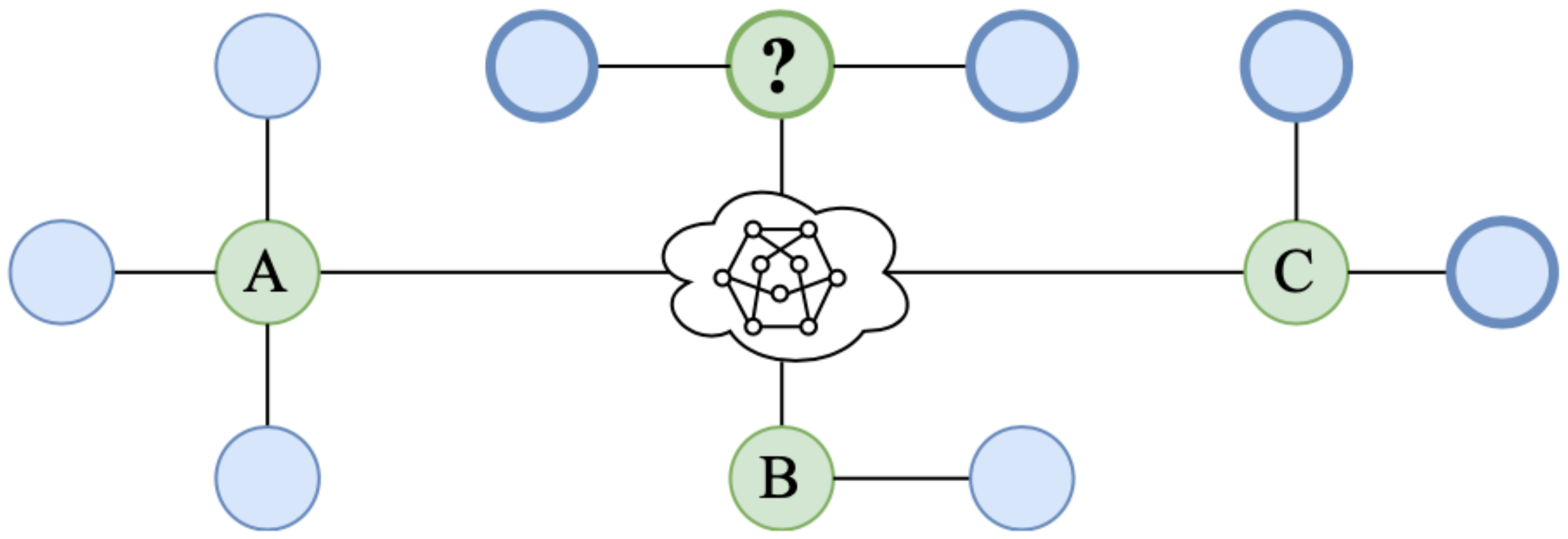


Figure 16: The NeighborsMatch problem. The correct output label for the depicted graph is C.[7]

Oversquashing

Empirical Results

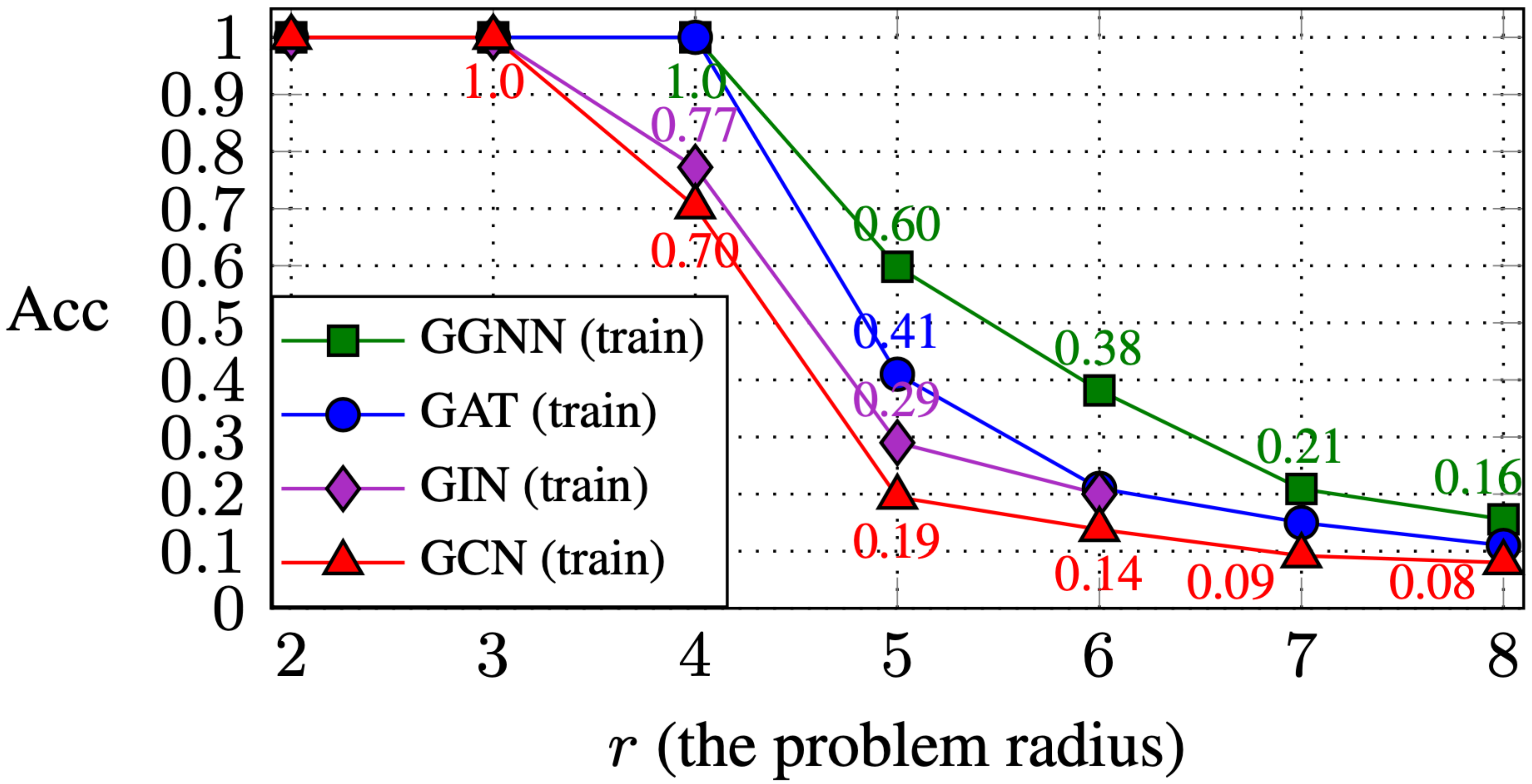


Figure 17: Performance of different GNNs on the NeighborsMatch problem. Underfitting (caused by oversquashing) can be observed from a problem radius of $r = 4$.[7]

Conclusion

What we covered

- Hierarchy of different GNN classes
- Anonymous GNNs are (at most) equivalent to the WL isomorphism test w.r.t. graph isomorphism
- Connection between GNNs and models of distributed computation
- Requirements of Turing completeness of GNNs with unique vertex IDs
- Limitations based on depth and width, and, more generally, communication capacity
- The problem of oversquashing in deep GNNs and problems with a large radius

References

(1) Xu, Keyulu, et al. 'How Powerful Are Graph Neural Networks?' *ArXiv:1810.00826 [Cs, Stat]*, Feb. 2019. *arXiv.org*, http://arxiv.org/abs/1810.00826.

(2) Sato, Ryoma, et al. 'Approximation Ratios of Graph Neural Networks for Combinatorial Problems'. *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. *Neural Information Processing Systems*, https://proceedings.neurips.cc/paper/2019/hash/635440afdfc39fe37995fed127d7df4f-Abstract.html.

(3) Huang, Ningyuan, and Soledad Villar. 'A Short Tutorial on The Weisfeiler-Lehman Test And Its Variants'. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2021, pp. 8533–37. *arXiv.org*, https://doi.org/10.1109/ICASSP39728.2021.9413523.

(4) Hella, Lauri, et al. 'Weak Models of Distributed Computing, with Connections to Modal Logic'. *Distributed Computing*, vol. 28, 2012, https://doi.org/10.1145/2332432.2332466.

(5) Loukas, Andreas, 'What Graph Neural Networks Cannot Learn: Depth vs Width'. *International Conference on Learning Representations*, 2020, https://openreview.net/forum?id=B1l2bp4YwS.

(6) Loukas, Andreas. 'How Hard Is to Distinguish Graphs with Graph Neural Networks?' *ArXiv:2005.06649 [Cs, Stat]*, Oct. 2020. *arXiv.org*, http://arxiv.org/abs/2005.06649.

(7) Alon, Uri, and Eran Yahav. 'On the Bottleneck of Graph Neural Networks and Its Practical Implications'. *International Conference on Learning Representations*, 2021, https://openreview.net/forum?id=i80OPhOCVH2.