

Seminar Series on Graph Neural Networks 02

On the Representational Power of GNNs

Yong-Min Shin

School of Mathematics and Computing (Computational Science and Engineering)

Yonsei University

2025.04.07



수학계산학부(계산과학공학)

School of Mathematics and Computing
(Computational Science and Engineering)



광주과학기술원

Gwangju Institute of Science and Technology

Towards application of graph neural networks

Towards efficient graph learning

Explainable graph neural networks

Fundamental topics on graph neural networks

On the representational power of graph neural networks

A graph signal processing viewpoint of graph neural networks

From label propagation to graph neural networks

On the problem of oversmoothing and oversquashing

Introduction to graph mining and graph neural networks
(Basic overview to kick things off)



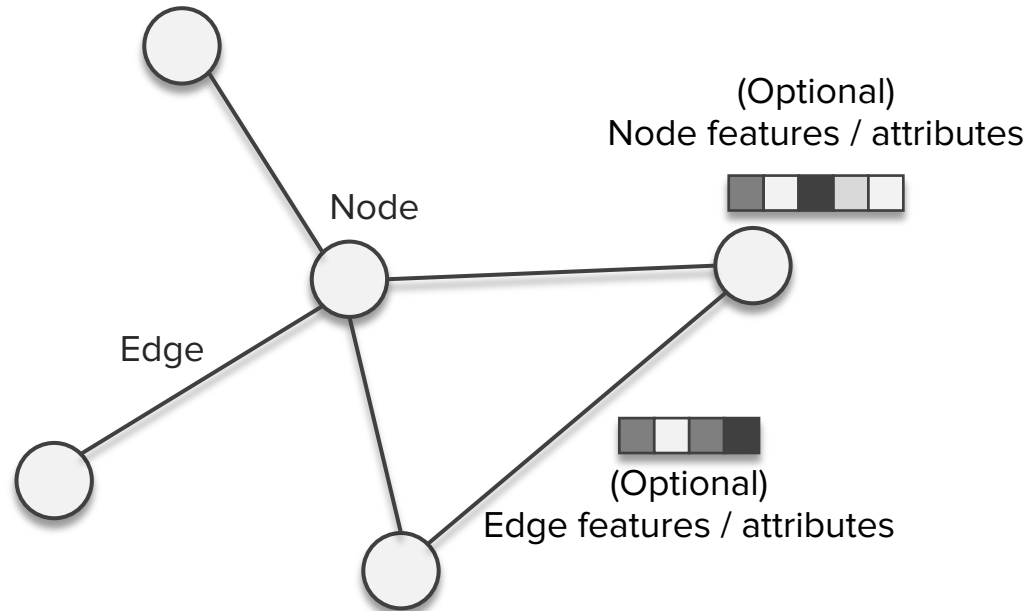
1. Understanding of **what makes two graphs the 'same'**
2. Understanding of the **Weisfeiler-Lehman isomorphism test**
3. Understanding the **connection** between the WL test and message-passing
4. In-depth understanding of (Xu et al., ICLR 2019) and (Morris et al., AAAI 2019)

*Today's topic is more relevant on chemical datasets, where the model needs to extract as much information as possible from the given graph structure.

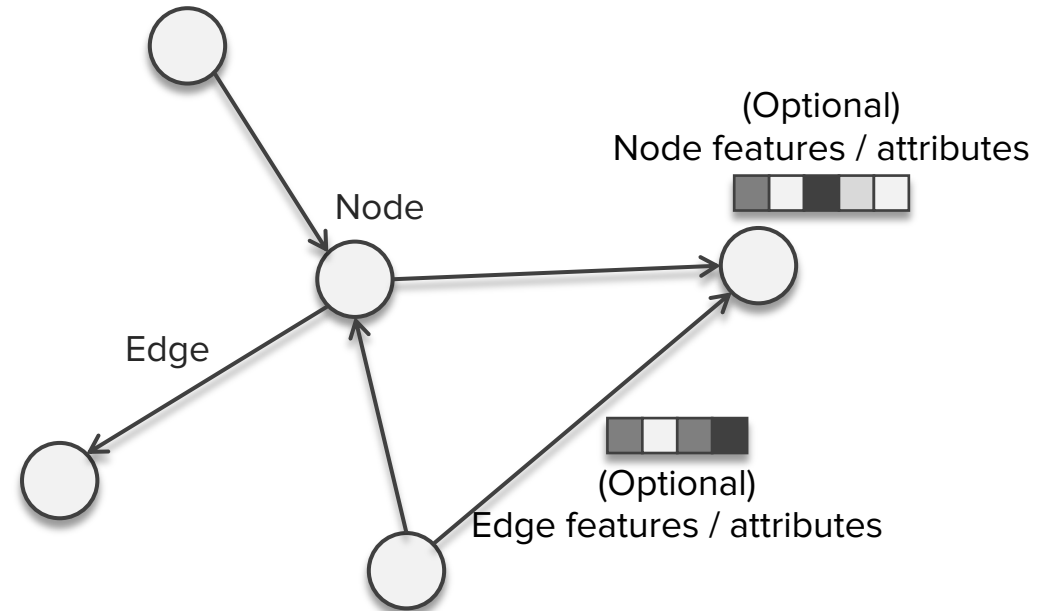
What makes two graphs the ‘same’?

(Revisit) Graphs as an abstract datatype

Graphs are an abstract type of data where nodes (entities) are **connected** by edges (connections)



Undirected graph

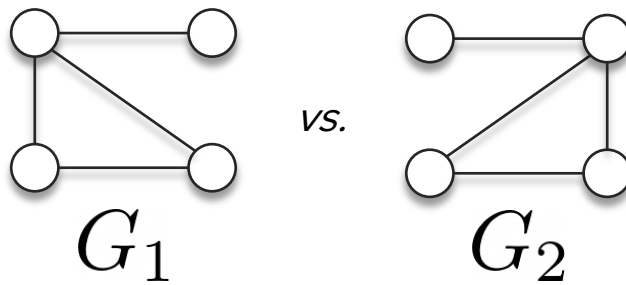


Directed graph

For now, let's assume we do not consider node / edge features.

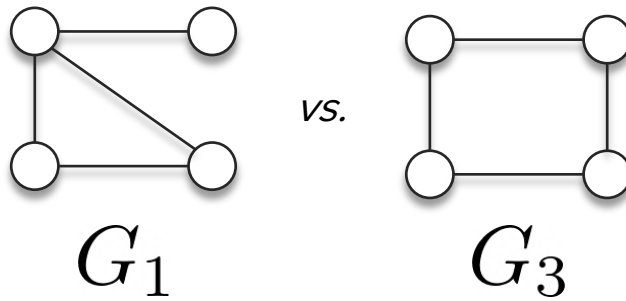
Only looking at the 'graph structure' (roughly speaking, connection patterns), how do we determine whether two graphs are the same?

Example 1

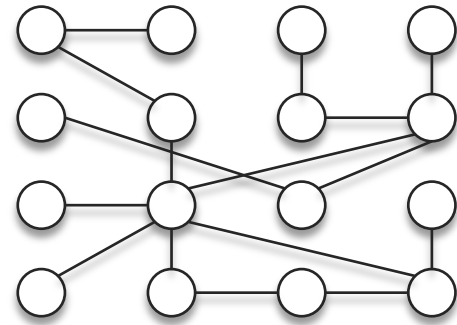


Example 2

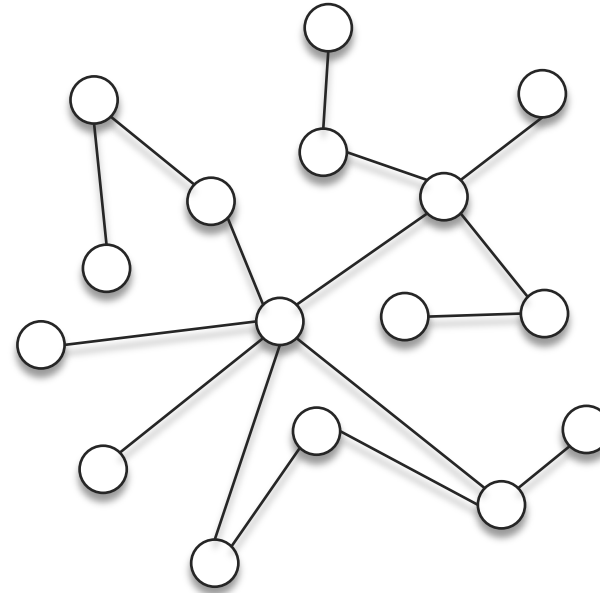
7



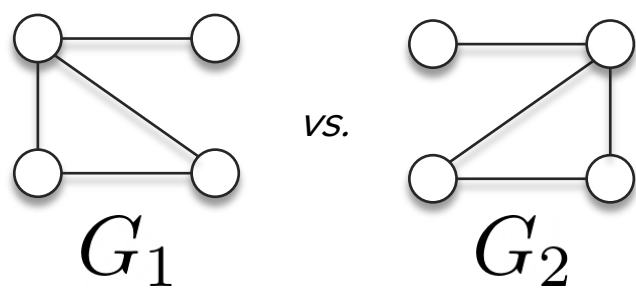
Example 3

 G_4

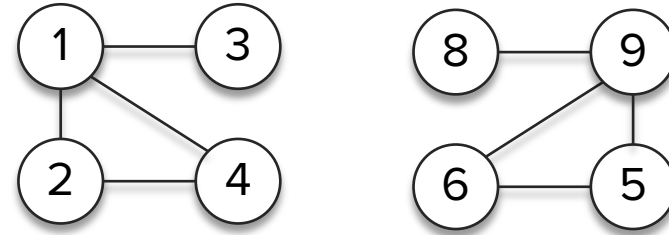
vs.

 G_5

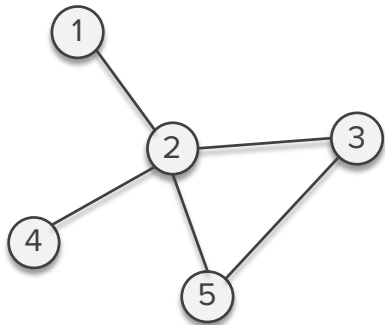
Isomorphism (a fancy word for identical graphs)



Whatever the definition of 'isomorphism' is,
it must not care about node orderings



We say that two graphs G and H are *isomorphic* if there exists an edge preserving bijection $\varphi : V(G) \rightarrow V(H)$, i.e., (u, v) is in $E(G)$ if and only if $(\varphi(u), \varphi(v))$ is in $E(H)$.



Assign arbitrary node ordering

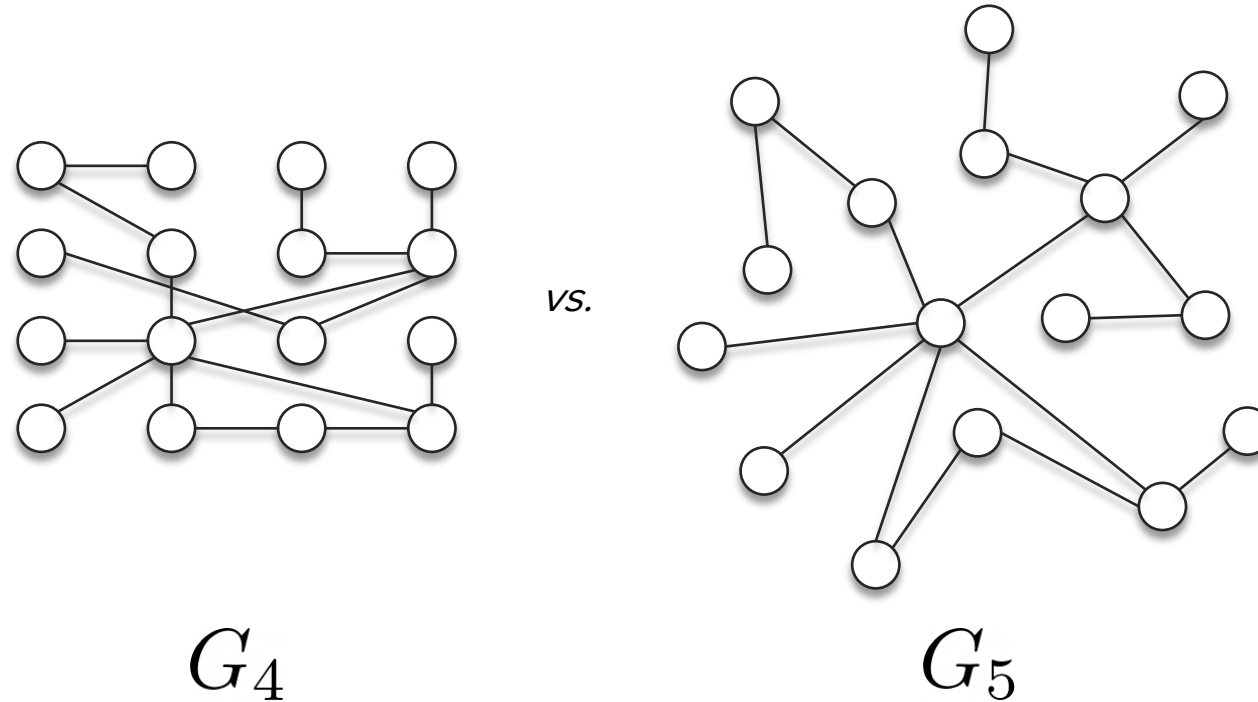
- **Graphs with canonical node ordering is not common**
- Related research topic: Positional encoding of nodes
(As an example, see [1])

This means, G_1 and G_2 are **isomorphic** since we can find a bijection of:

3 – 8
1 – 9
4 – 6
2 – 5

and according to this node mapping, the edge set from G_1 exactly translates to G_2 .

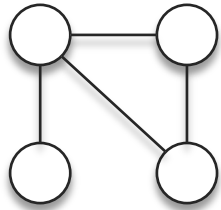
Remember, there are no 'correct' node ordering.



*The problem of graph isomorphism testing is suspected to be *NP-hard [2], [3]*

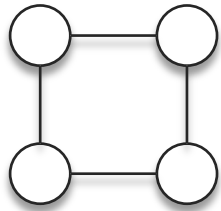
- Probably no exact (deterministic) polynomial-time algorithmic solutions
- **WL isomorphism test:** A heuristic algorithm to test isomorphism

Understanding the WL-isomorphism test



Graph 1

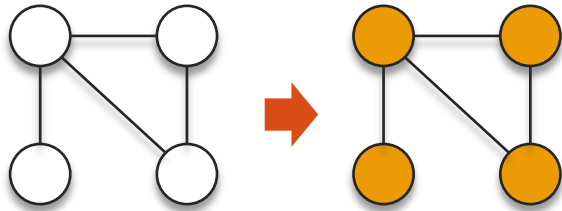
Q. Is there a systematic (heuristic) method that can “mostly” identify isomorphic graphs?



Graph 2

One iteration of the WL-isomorphism test [1], [2]

13

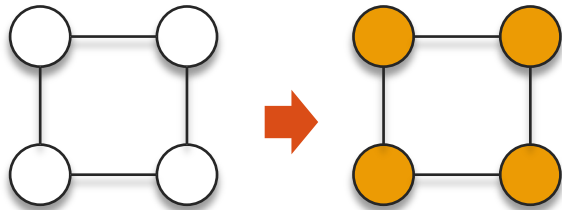


Graph 1

(Initial iteration only)

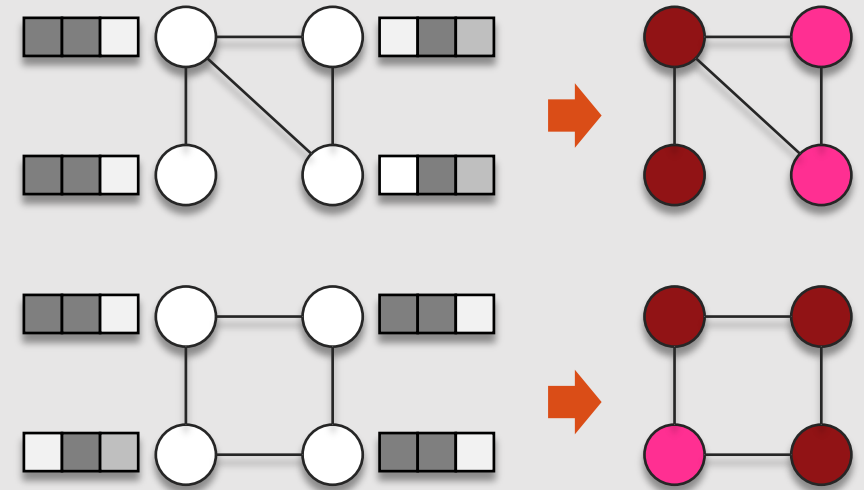
1

Color nodes †appropriately



Graph 2

Graphs with node features: Also appropriately

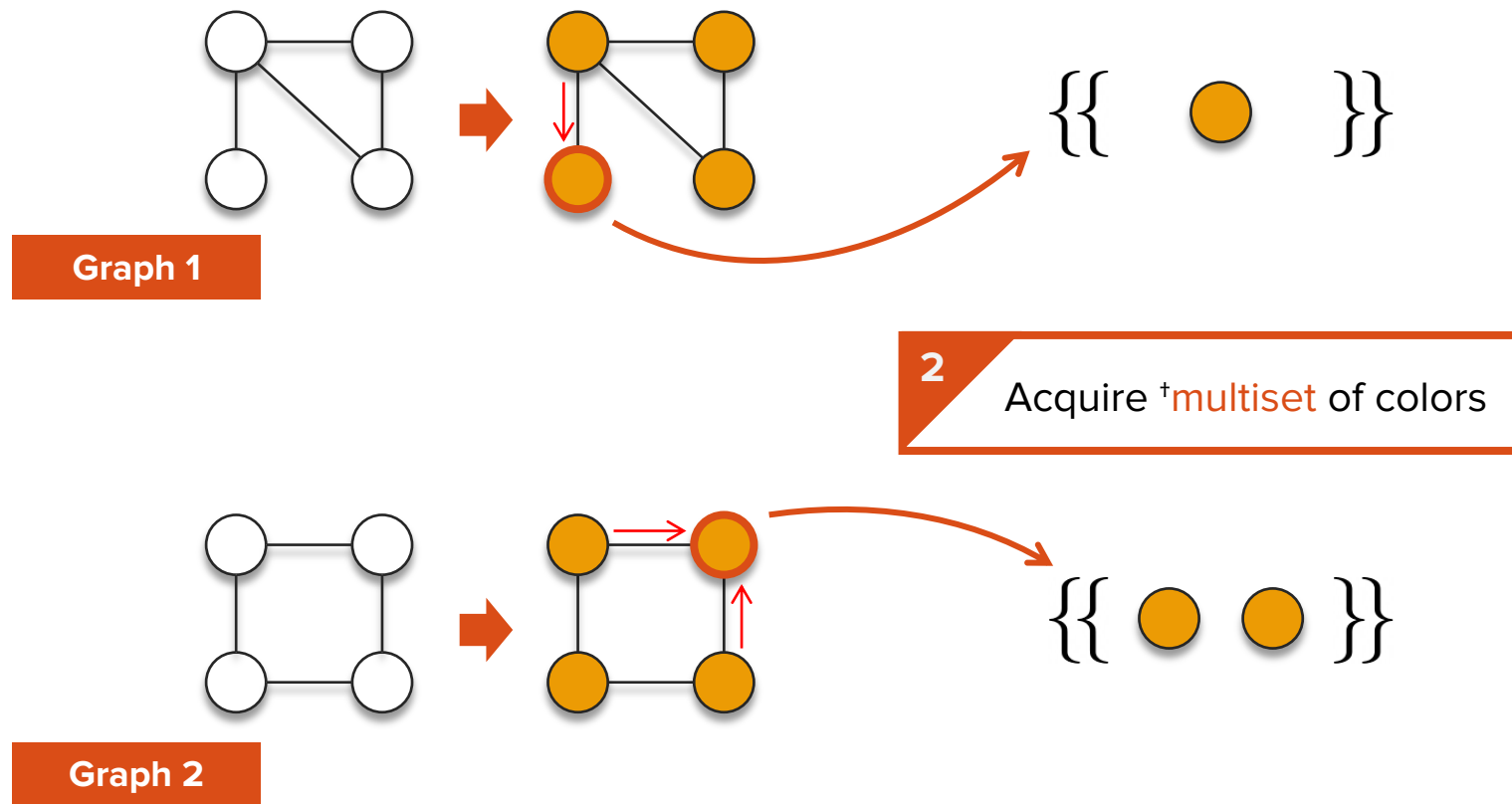


†As suggested by [4], color node according to the node degree. Or just start with a uniform coloring

[4] Shervashidze et al., “Weisfeiler-Lehman Graph Kernels”, J. Mach. Learn. Res. (2011)

[5] Morris et al., “Weisfeiler and Leman go Machine Learning: The Story so far”, arXiv (2021)

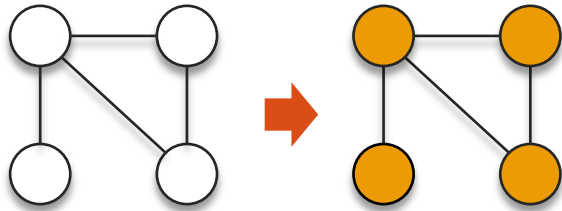
One iteration of the WL-isomorphism test [1], [2]



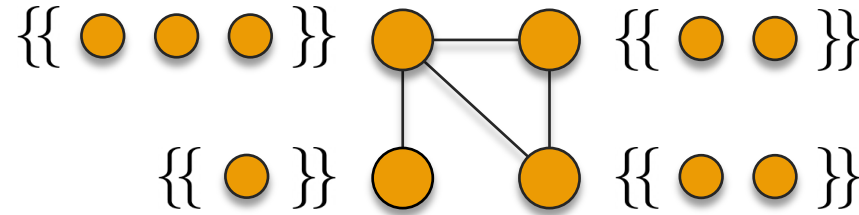
[†]Multiset is a set that allows multiple duplicates of elements

[4] Shervashidze et al., "Weisfeiler-Lehman Graph Kernels", J. Mach. Learn. Res. (2011)

[5] Morris et al., "Weisfeiler and Leman go Machine Learning: The Story so far", arXiv (2021)

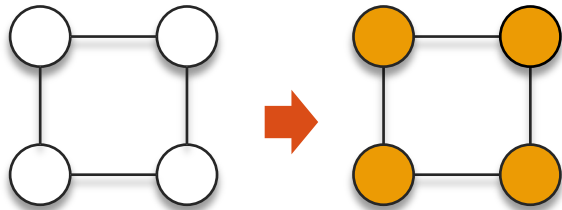


Graph 1

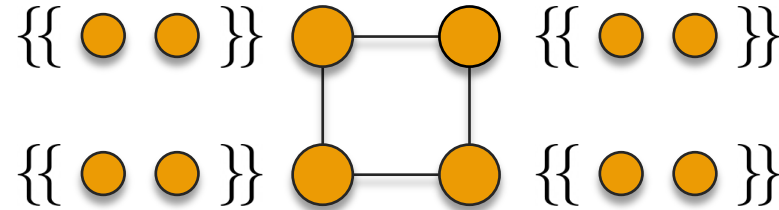


2

Acquire [†]multiset of colors



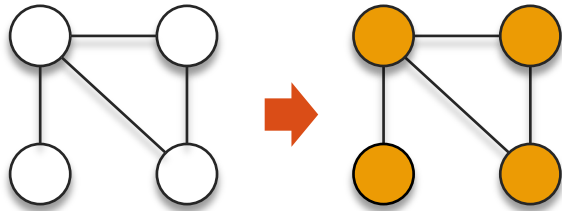
Graph 2



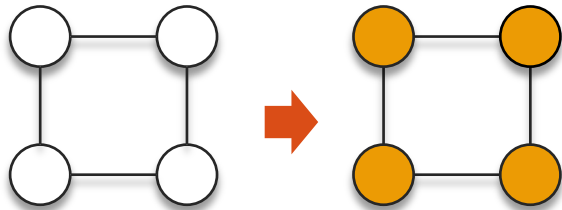
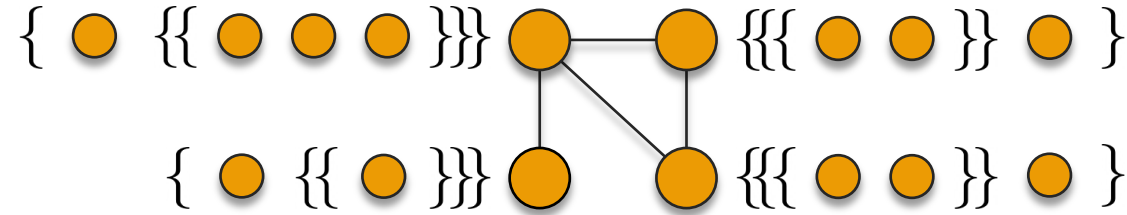
[†]Multiset is a set that allows multiple duplicates of elements

[4] Shervashidze et al., "Weisfeiler-Lehman Graph Kernels", J. Mach. Learn. Res. (2011)

[5] Morris et al., "Weisfeiler and Leman go Machine Learning: The Story so far", arXiv (2021)



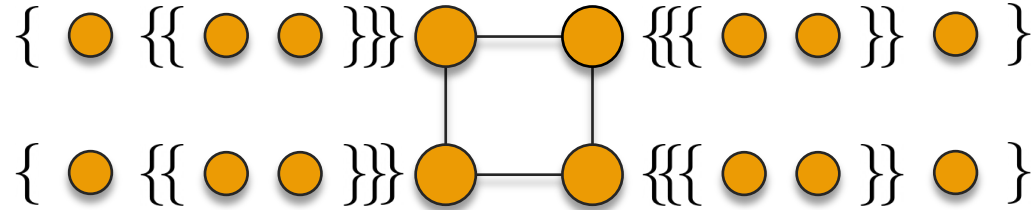
Graph 1



Graph 2

3

Make a set by including self



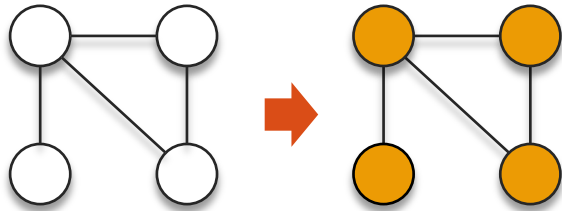
*Multiset is a set that allows multiple duplicates of elements

[4] Shervashidze et al., "Weisfeiler-Lehman Graph Kernels", J. Mach. Learn. Res. (2011)

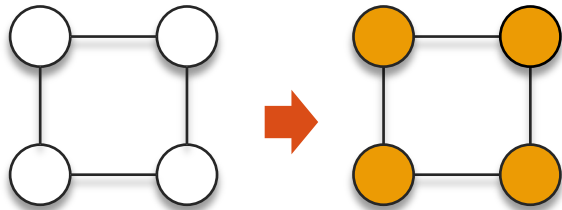
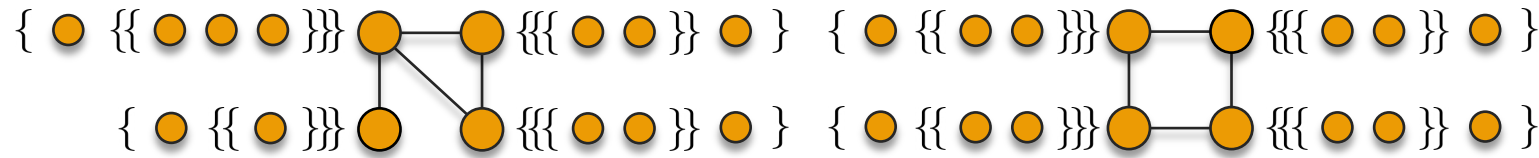
[5] Morris et al., "Weisfeiler and Leman go Machine Learning: The Story so far", arXiv (2021)

One iteration of the WL-isomorphism test [1], [2]

17



Graph 1



Graph 2

4

Map each set to a **new color** by a [†]**bijective** function

$$\text{purple node} \leftarrow \text{hash}(\{\text{orange node} \{\{\text{orange node orange node}\}\}\})$$

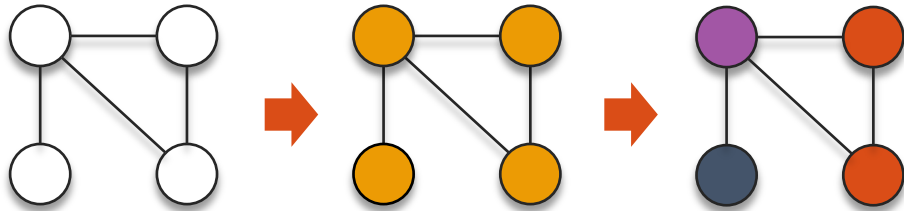
$$\text{dark blue node} \leftarrow \text{hash}(\{\text{orange node} \{\{\text{orange node}\}\}\})$$

$$\text{red node} \leftarrow \text{hash}(\{\text{orange node} \{\{\text{orange node orange}\}\}\})$$

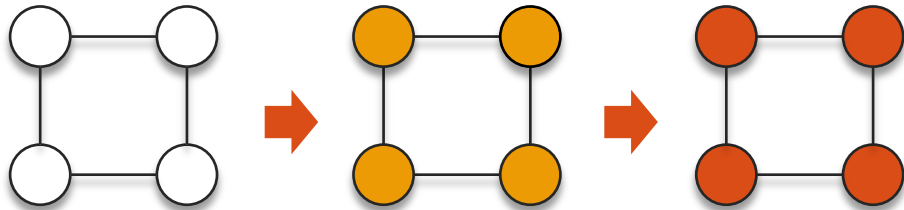
[†] At least injective. The function has multiple names, such as hashing functions, relabeling functions, etc.

[4] Shervashidze et al., “Weisfeiler-Lehman Graph Kernels”, J. Mach. Learn. Res. (2011)

[5] Morris et al., “Weisfeiler and Leman go Machine Learning: The Story so far”, arXiv (2021)



Graph 1



Graph 2

5

Get the colors of the next iteration

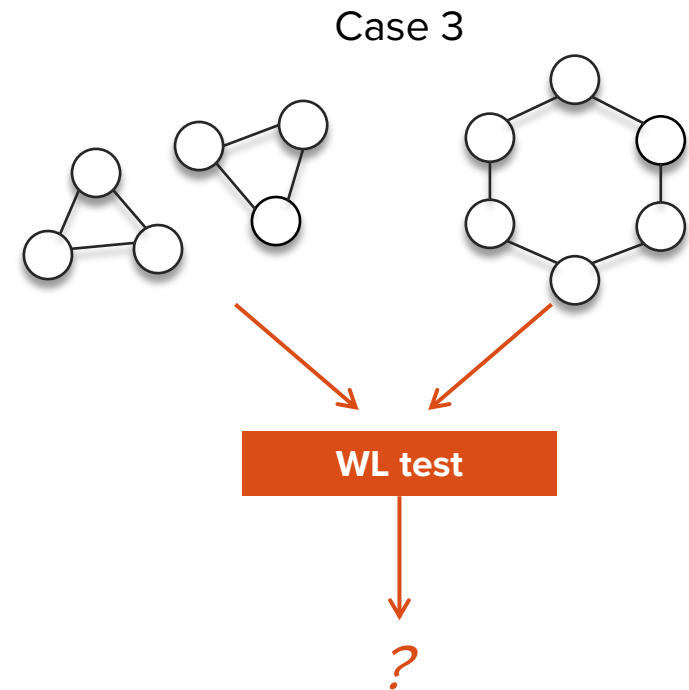
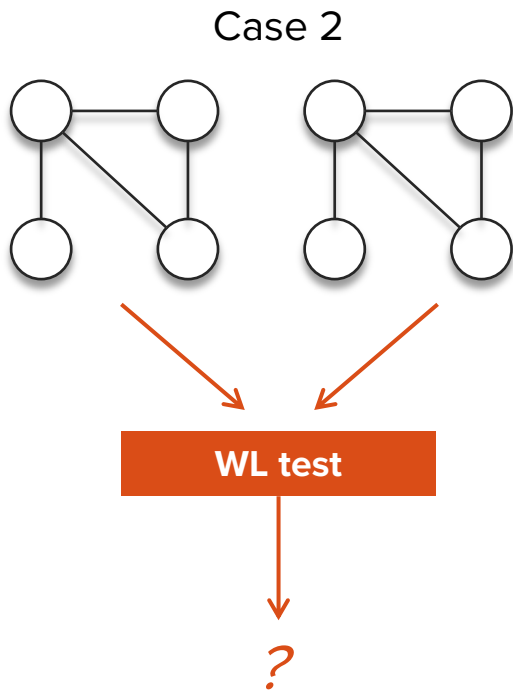
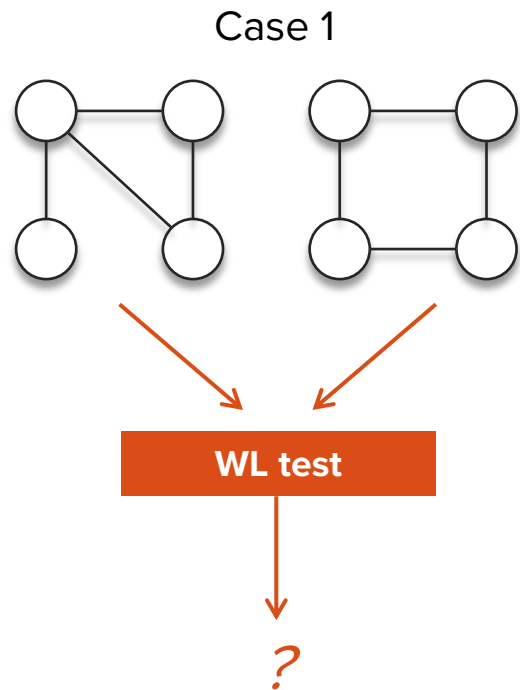
$$\text{purple} \leftarrow \text{hash}(\{ \text{yellow} \{ \{ \text{yellow} \text{yellow} \text{yellow} \} \} \})$$

$$\text{blue} \leftarrow \text{hash}(\{ \text{yellow} \{ \{ \text{yellow} \} \} \})$$

$$\text{red} \leftarrow \text{hash}(\{ \text{yellow} \{ \{ \text{yellow} \text{yellow} \} \} \})$$

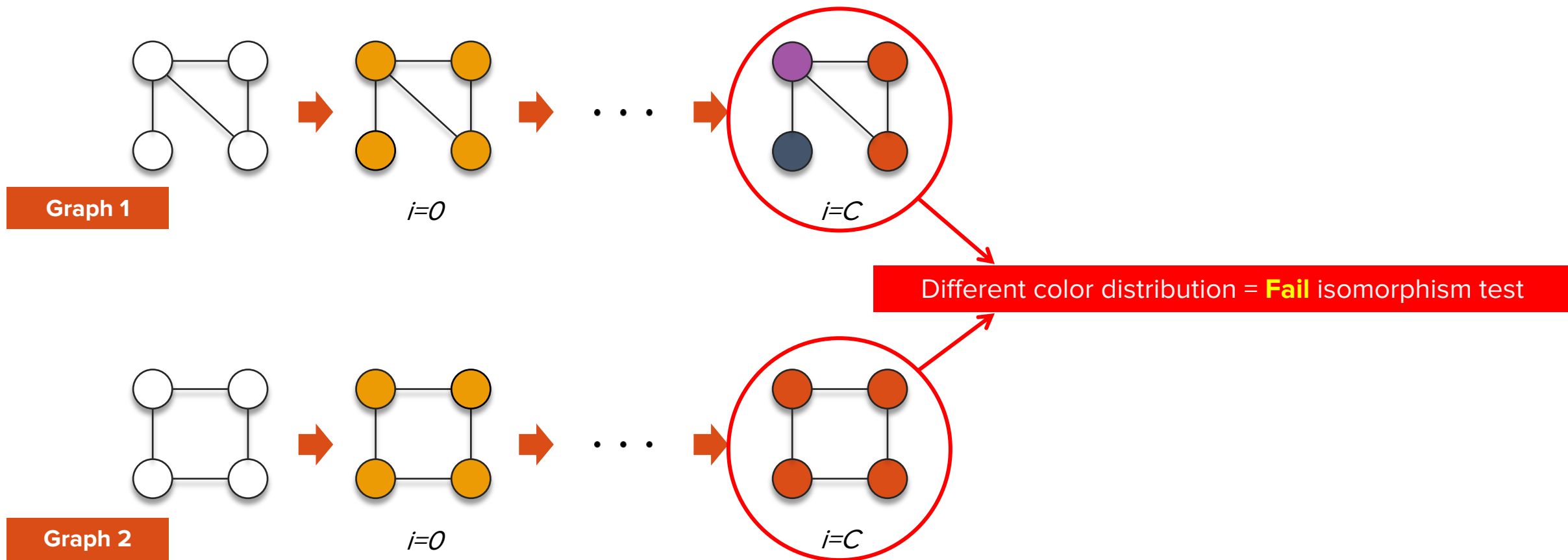
WL-isomorphism test: Three example cases

19



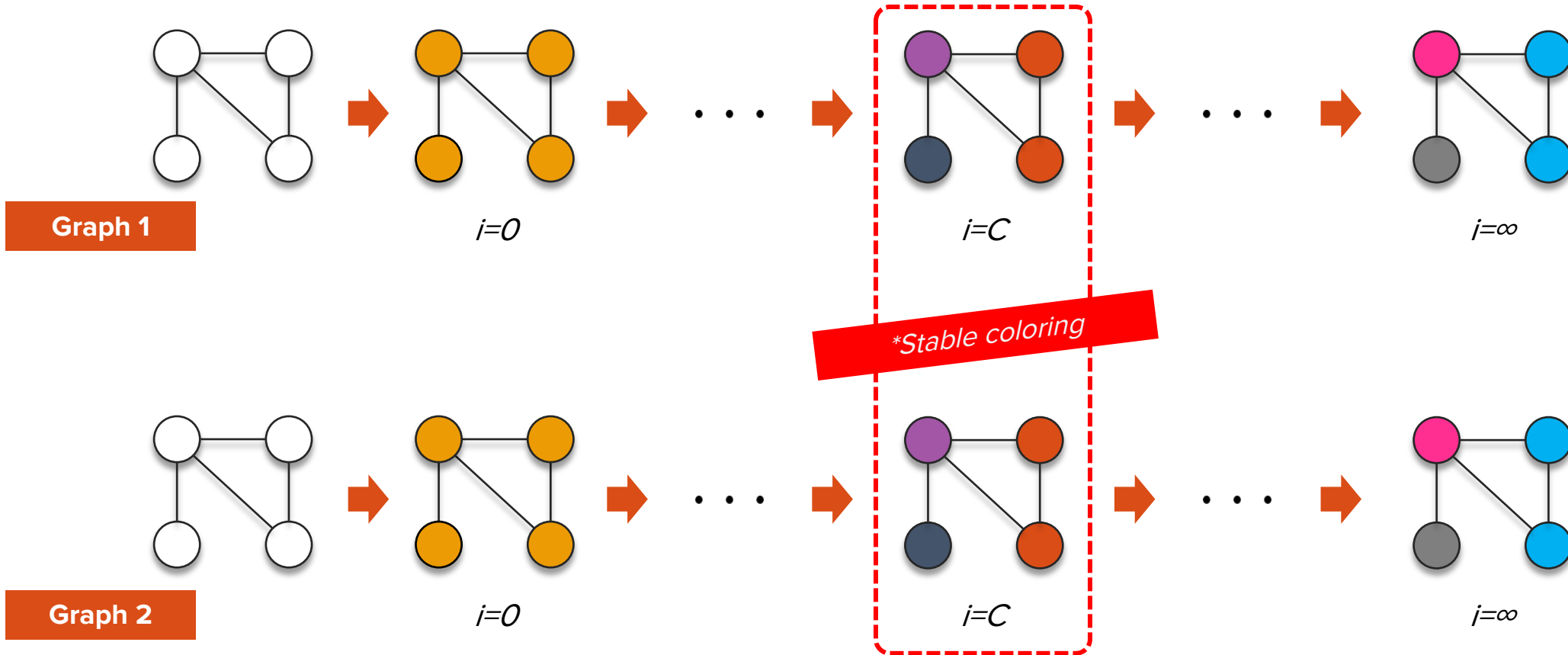
WL-isomorphism test: Three example cases

Conclusion of case 1



WL-isomorphism test: Three example cases

Conclusion of case 2

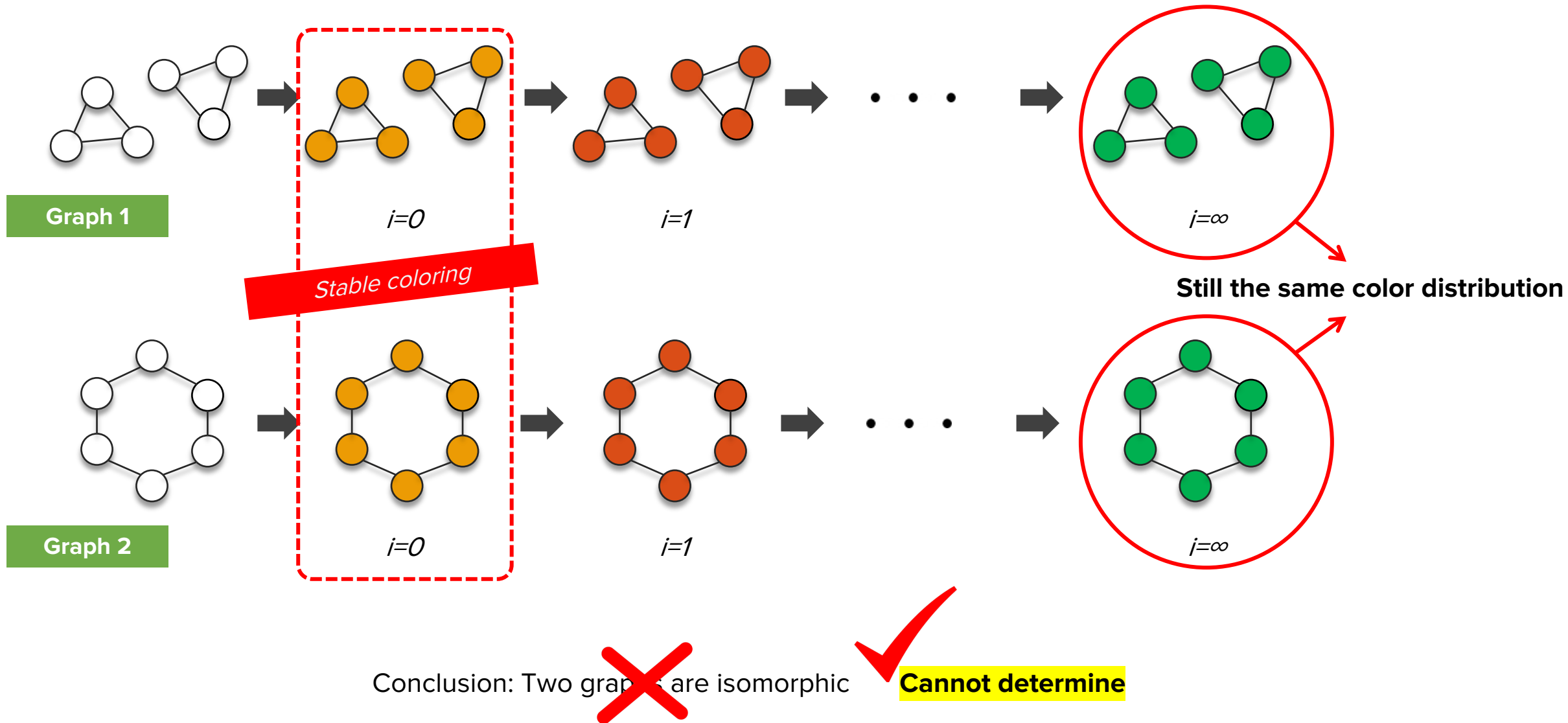


Conclusion: Two graphs are isomorphic ..?

* We do not actually need to run the iteration to the end of time: If color distributions remain unchanged for two consecutive iterations, you already reached stable coloring (hint: Use induction). Also, C is bounded by $\max(|\text{Graph 1}|, |\text{Graph 2}|)$ (see [5]).

WL-isomorphism test: Three example cases

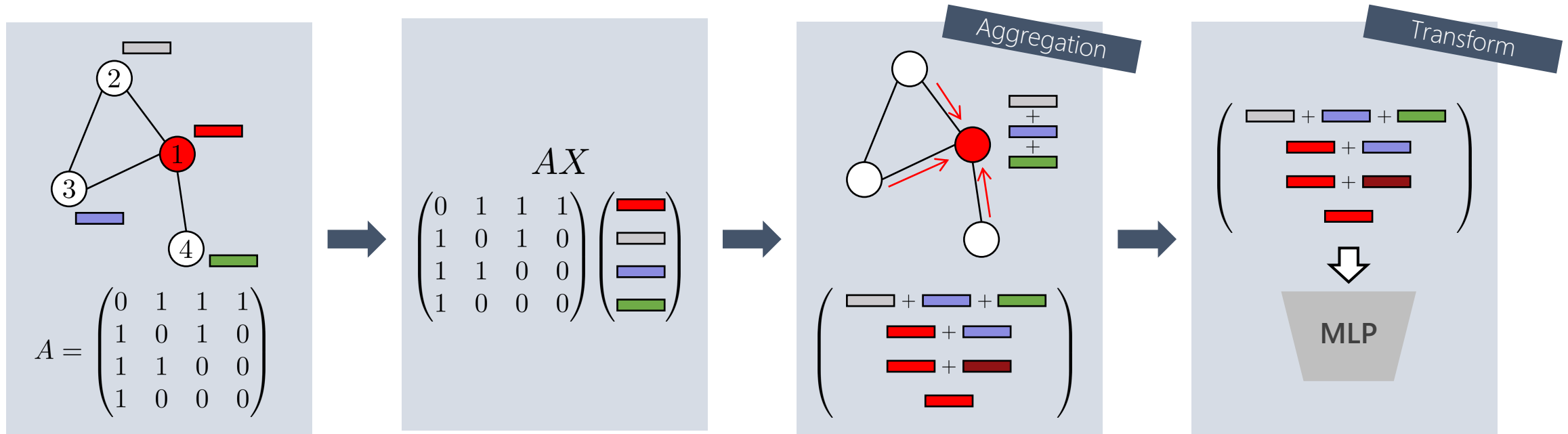
Conclusion of case 3



Understanding the connection between the WL test and message-passing

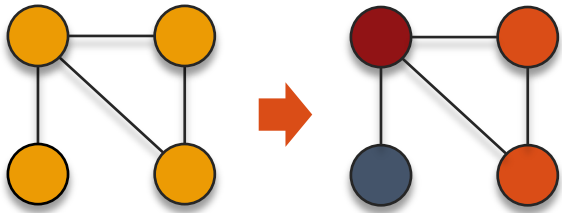
(Recap) Message-passing framework in GNNs

Aggregate and Transform



Relation between WL and GNNs

“Color refinement” in WL

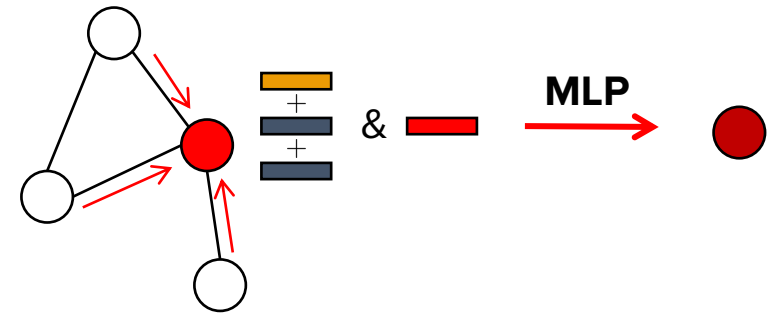


$$\text{purple circle} \leftarrow \text{hash}(\{ \text{yellow circle} \{ \{ \text{yellow circle} \text{ yellow circle} \text{ yellow circle} \} \} \})$$

$$\text{dark blue circle} \leftarrow \text{hash}(\{ \text{yellow circle} \{ \{ \text{yellow circle} \} \} \})$$

$$\text{orange circle} \leftarrow \text{hash}(\{ \text{yellow circle} \{ \{ \text{yellow circle} \text{ yellow circle} \} \} \})$$

Message passing in GNNs



1. Aggregate

2. Transform

$$\mathbf{h}_u = \phi \left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v) \right)$$

Can you see the similarity?

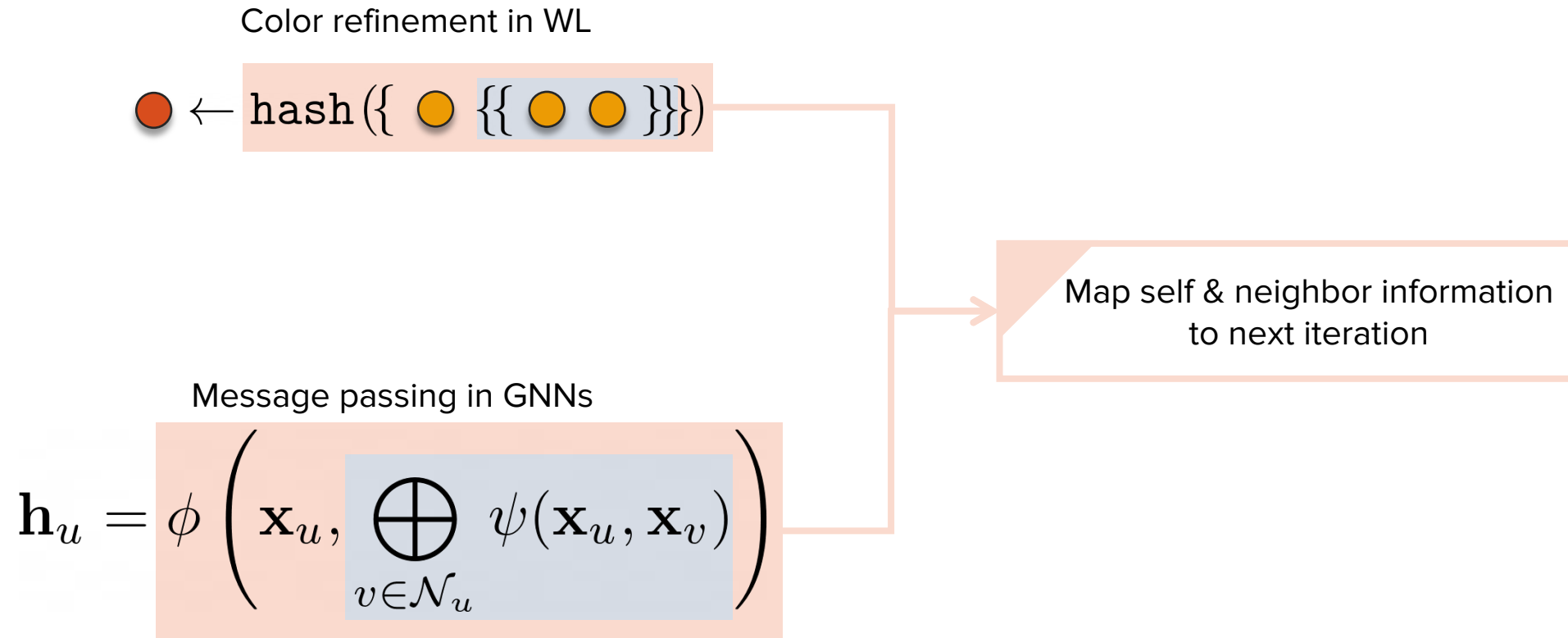
Color refinement in WL

$$\text{red circle} \leftarrow \text{hash}(\{ \text{yellow circle} \{ \{ \text{yellow circle} \text{yellow circle} \} \} \})$$

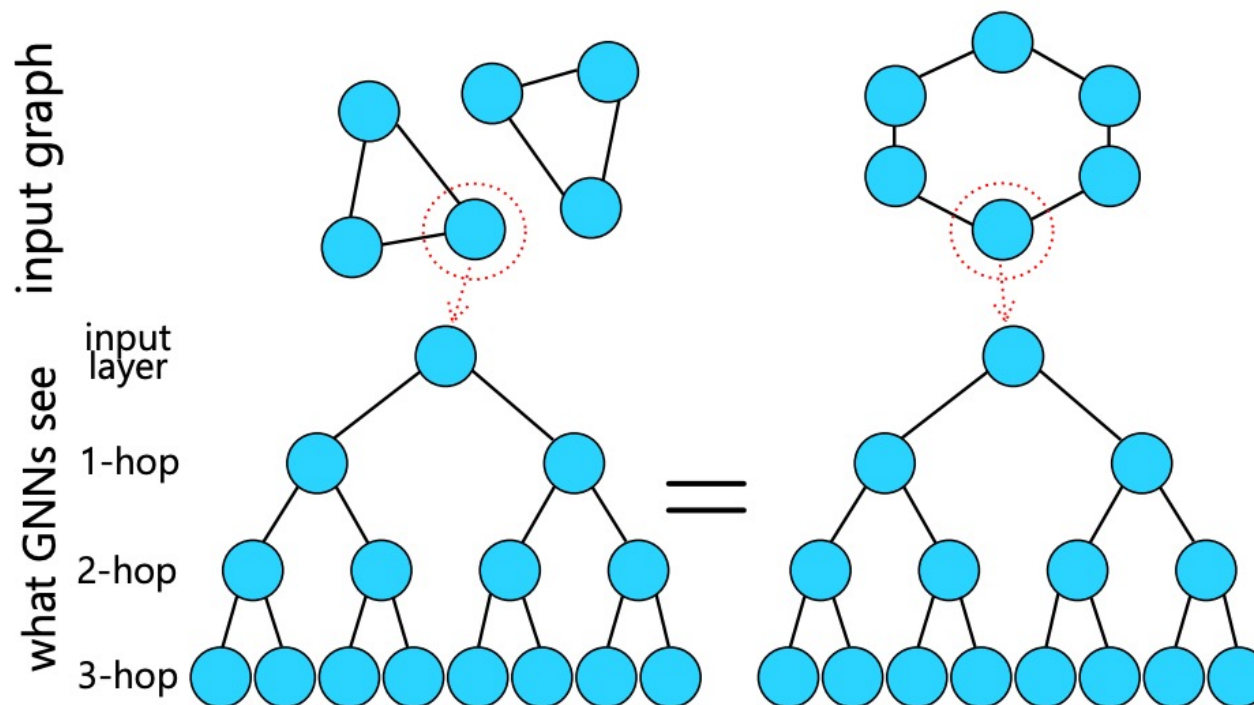
Collect neighbor information

Message passing in GNNs

$$\mathbf{h}_u = \phi \left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v) \right)$$



Revisiting Case 3



The same intuition can also be derived from the “computational tree” point of view [6].

Consequences of GNN's ability to differentiate graphs

Color refinement in WL

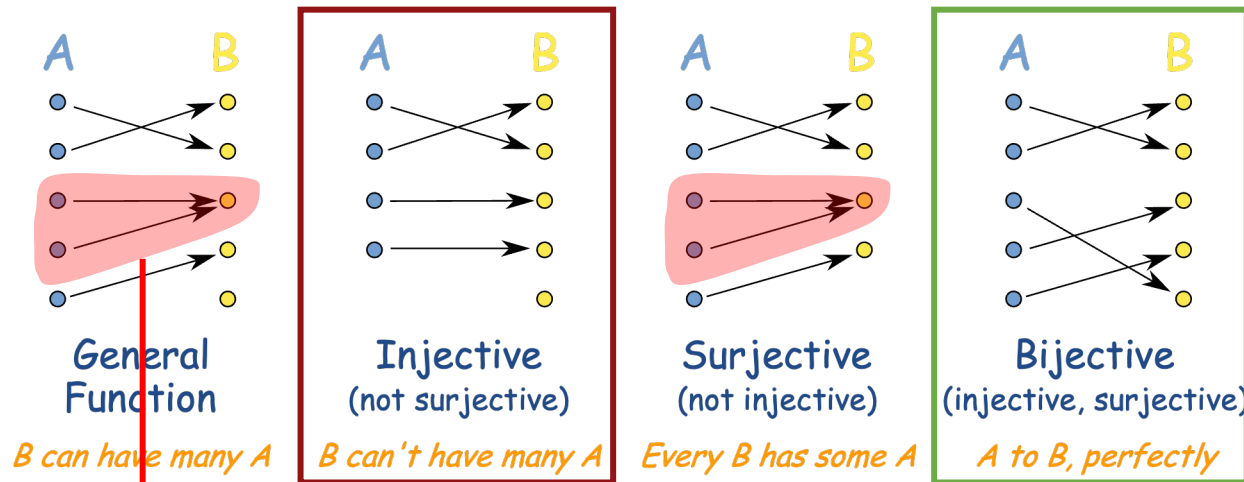
$$\text{red circle} \leftarrow \text{hash}(\{\text{orange circle} \{\{\text{orange circle} \text{ orange circle}\}\}\})$$

- **hash**: Fixed **bijective** function (at least **injective**)

Message passing in GNNs

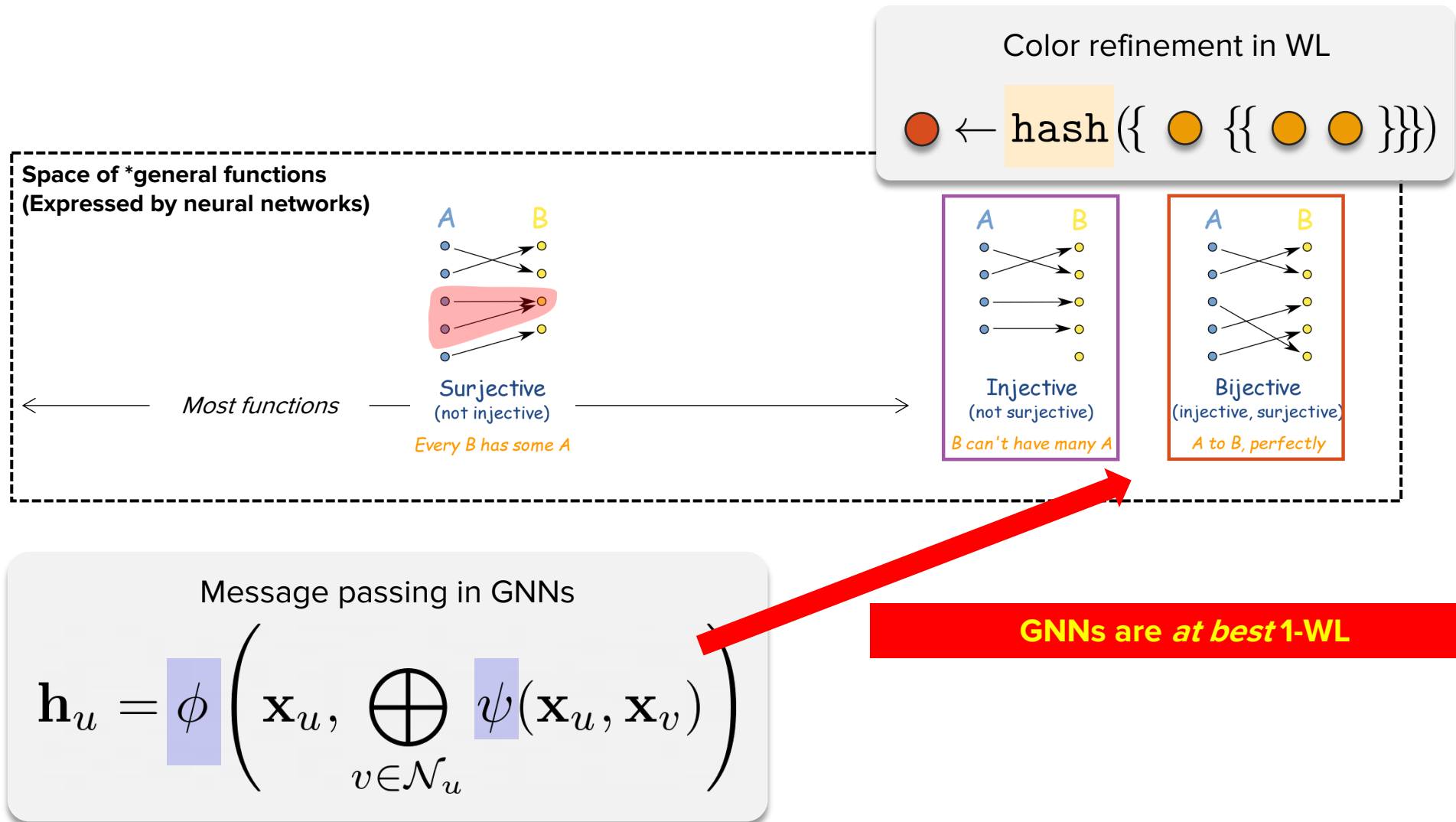
$$\mathbf{h}_u = \phi \left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v) \right)$$

- ϕ, ψ : A neural network (Learned from data)
- (Probably) *Not bijective nor injective*



Loss of expressive power: Cannot distinguish some elements

Consequences of GNN's ability to differentiate graphs



In-depth understanding of (Xu et al., ICLR 2019) and (Morris et al., AAAI 2019)

GNNs cannot exceed WL in terms of its expressivity

Theorem [Morris et al., 2019, Xu et al., 2019] (informal)

If the 1-WL test cannot distinguish two graphs, then any GNNs also cannot distinguish them.

If GNNs can distinguish two graphs, the 1-WL test can also distinguish them.

In other words, the expressive power of GNNs is capped by 1-WL.

Color refinement in 1-WL

$$\text{red circle} \leftarrow \text{hash}(\{ \text{orange circle} \{ \{ \text{orange circle} \text{orange circle} \} \} \})$$

$$\geq$$

Message passing in GNNs

$$\mathbf{h}_u = \phi \left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v) \right)$$

GNNs cannot exceed WL in terms of its expressivity

Proof of existence

Theorem (informal)

There exists weight parameters of GNN such that, expressivity of GNNs **exactly match** 1-WL test.

How to go beyond?

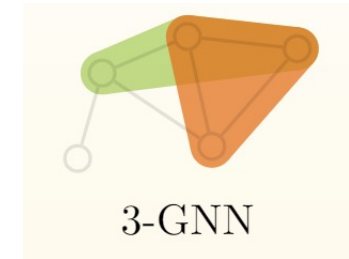
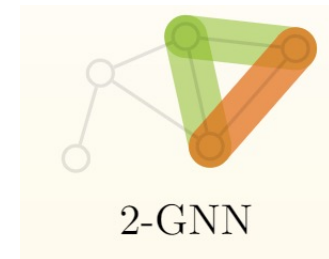
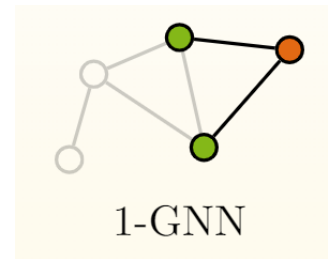
Problem: GNNs are bound by 1-dim WL-test

Solution: Make GNNs based on **k -dim WL-test**
($k > 1$)

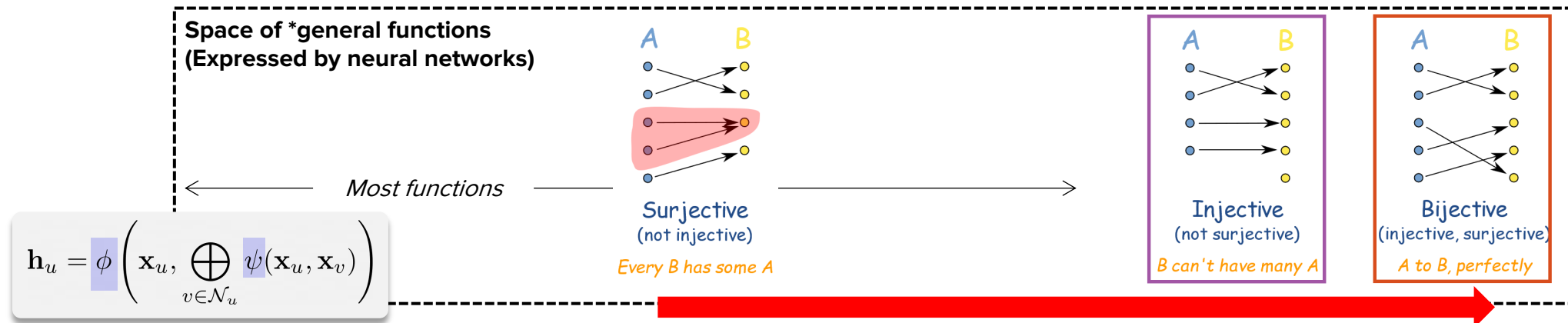
Theorem 2. Let (G, l) be a labeled graph. Then for all $t \geq 0$ there exists a sequence of weights $\mathbf{W}^{(t)}$, and a 1-GNN architecture such that

$$c_i^{(t)} \equiv f^{(t)}.$$

Hence, in the light of the above results, 1-GNNs may be viewed as an extension of the 1-WL which in principle have the same power but are more flexible in their ability to adapt to the learning task at hand and are able to handle continuous node features.



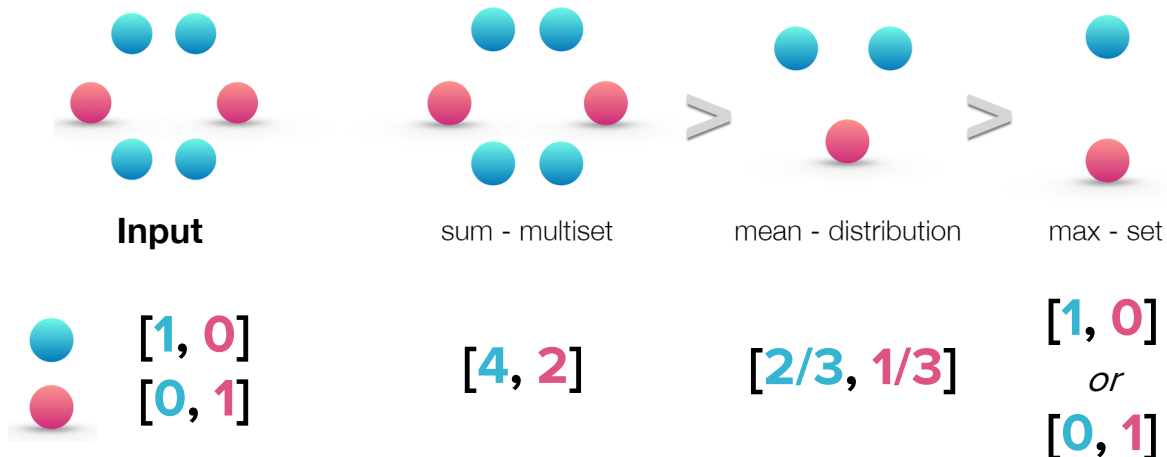
GNNs cannot exceed WL in terms of its expressivity



Q. What design choices are needed to make the function *injective as possible?

1. Use summation for aggregation

2. Use at least 2 layers of MLP



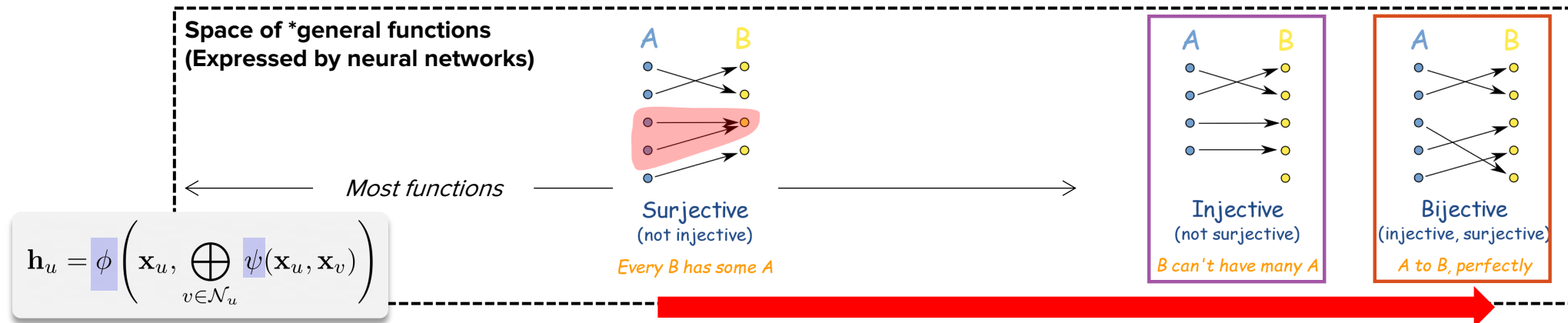
Theorem [Xu et al., 2019] (informal)

One-layer ReLU MLPs are *not* injective.

* Does not necessarily mean the resulting neural network is injective.

For injectivity in neural networks, see Puthawala et al., "Globally Injective ReLU Networks", J. Mach. Learn. Res. (2020)

GNNs cannot exceed WL in terms of its expressivity



Q. What design choices are needed to make the function *injective as possible?

Graph Isomorphism Networks (GIN)

$$h_v^{(k)} = \text{MLP}^{(k)} \left(\left(1 + \epsilon^{(k)} \right) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} \right)$$

*In my experience, just setting epsilon as a non-learnable parameter with 0 value works fine

1. Defining graphs being 'identical' = isomorphism test
2. WL-isomorphism test: Heuristic that can be used for isomorphism, but not 100% work
3. Connections: GNN's message-passing and WL test, and GNN's limitations

Thank you!

Please feel free to ask any questions :)

jordan7186.github.io