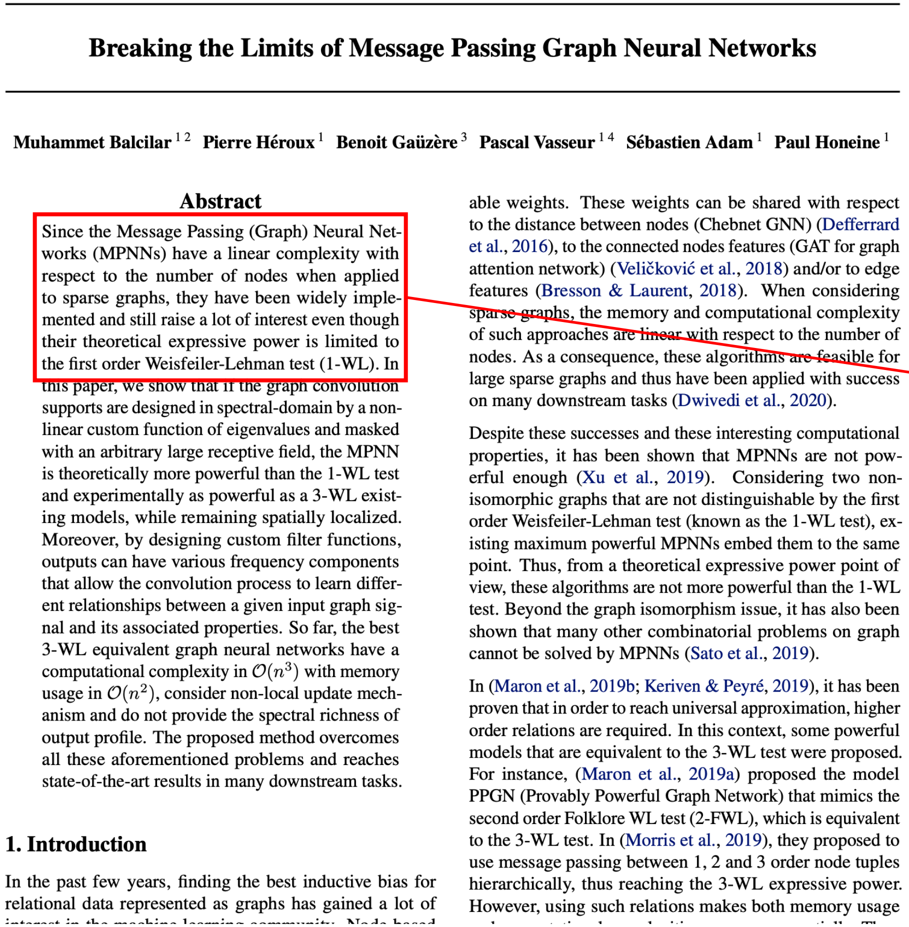
Towards understanding the connection between GNNs and the Weisfeiler-Lehman test

Yong-Min Shin
Materials intelligence lab

2026.01.15th 14:00
online event

Objective



Breaking the Limits of Message Passing Graph Neural Networks

Muhammet Balcilar[1 2] Pierre Héroux[1] Benoit Gaüzère[3] Pascal Vasseur[1 4] Sébastien Adam[1] Paul Honeine[1]

Abstract

Since the Message Passing (Graph) Neural Networks (MPNNs) have a linear complexity with respect to the number of nodes when applied to sparse graphs, they have been widely implemented and still raise a lot of interest even though their theoretical expressive power is limited to the first order Weisfeiler-Lehman test (1-WL). In this paper, we show that if the graph convolution supports are designed in spectral-domain by a non-linear custom function of eigenvalues and masked with an arbitrary large receptive field, the MPNN is theoretically more powerful than the 1-WL test and experimentally as powerful as a 3-WL existing models, while remaining spatially localized. Moreover, by designing custom filter functions, outputs can have various frequency components that allow the convolution process to learn different relationships between a given input graph signal and its associated properties. So far, the best 3-WL equivalent graph neural networks have a computational complexity in $\mathcal{O}(n^3)$ with memory usage in $\mathcal{O}(n^2)$, consider non-local update mechanism and do not provide the spectral richness of output profile. The proposed method overcomes all these aforementioned problems and reaches state-of-the-art results in many downstream tasks.

1. Introduction

In the past few years, finding the best inductive bias for relational data represented as graphs has gained a lot of interest in the machine learning community. Node-based

able weights. These weights can be shared with respect to the distance between nodes (Chebnet GNN) (Defferrard et al., 2016), to the connected nodes features (GAT for graph attention network) (Veličković et al., 2018) and/or to edge features (Bresson & Laurent, 2018). When considering sparse graphs, the memory and computational complexity of such approaches are linear with respect to the number of nodes. As a consequence, these algorithms are feasible for large sparse graphs and thus have been applied with success on many downstream tasks (Dwivedi et al., 2020).

Despite these successes and these interesting computational properties, it has been shown that MPNNs are not powerful enough (Xu et al., 2019). Considering two non-isomorphic graphs that are not distinguishable by the first order Weisfeiler-Lehman test (known as the 1-WL test), existing maximum powerful MPNNs embed them to the same point. Thus, from a theoretical expressive power point of view, these algorithms are not more powerful than the 1-WL test. Beyond the graph isomorphism issue, it has also been shown that many other combinatorial problems on graph cannot be solved by MPNNs (Sato et al., 2019).

In (Maron et al., 2019b; Keriven & Peyré, 2019), it has been proven that in order to reach universal approximation, higher order relations are required. In this context, some powerful models that are equivalent to the 3-WL test were proposed. For instance, (Maron et al., 2019a) proposed the model PPGN (Provably Powerful Graph Network) that mimics the second order Folklore WL test (2-FWL), which is equivalent to the 3-WL test. In (Morris et al., 2019), they proposed to use message passing between 1, 2 and 3 order node tuples hierarchically, thus reaching the 3-WL expressive power. However, using such relations makes both memory usage

"Since the **Message Passing (Graph) Neural Networks (MPNNs)** have ...
...even though their theoretical expressive power is limited to the first order **Weisfeiler-Lehman test (1-WL)**."

Objectives (Detailed)

Very introductory stuff {

1. Understanding graphs as a datatype
2. Understanding the general architecture of graph neural networks
3. Understanding of **what makes two graphs the 'same'**
4. Understanding of the **Weisfeiler-Lehman isomorphism test**
5. Understanding the **connection** between the WL test and message-passing
6. 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.
**I will try to refrain from mathematical jargon as much as possible, and only use them in order to summarize the conceptual content.

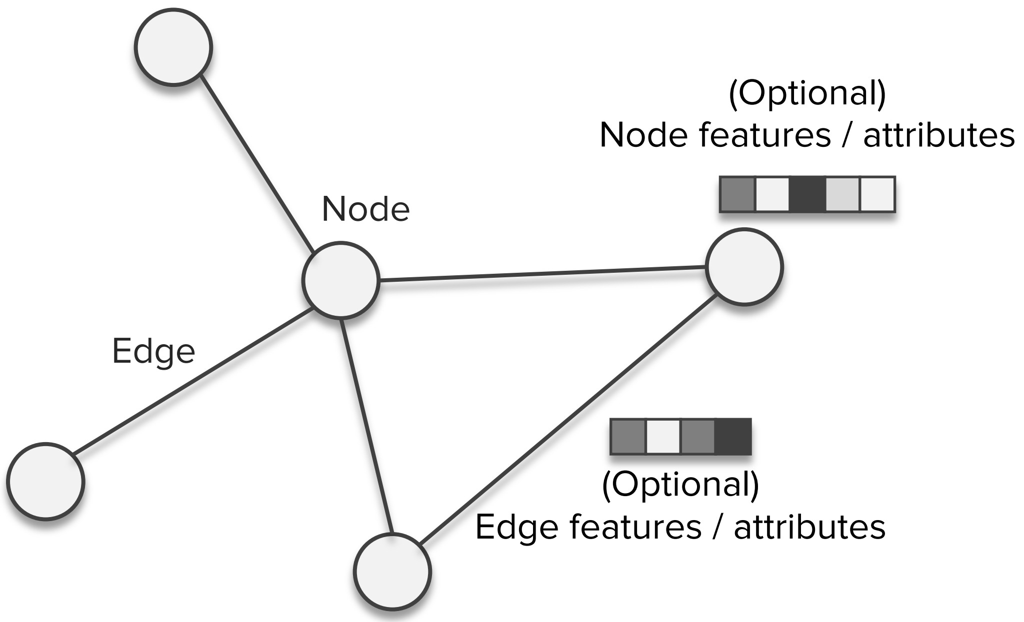
Buildup

Understanding graphs

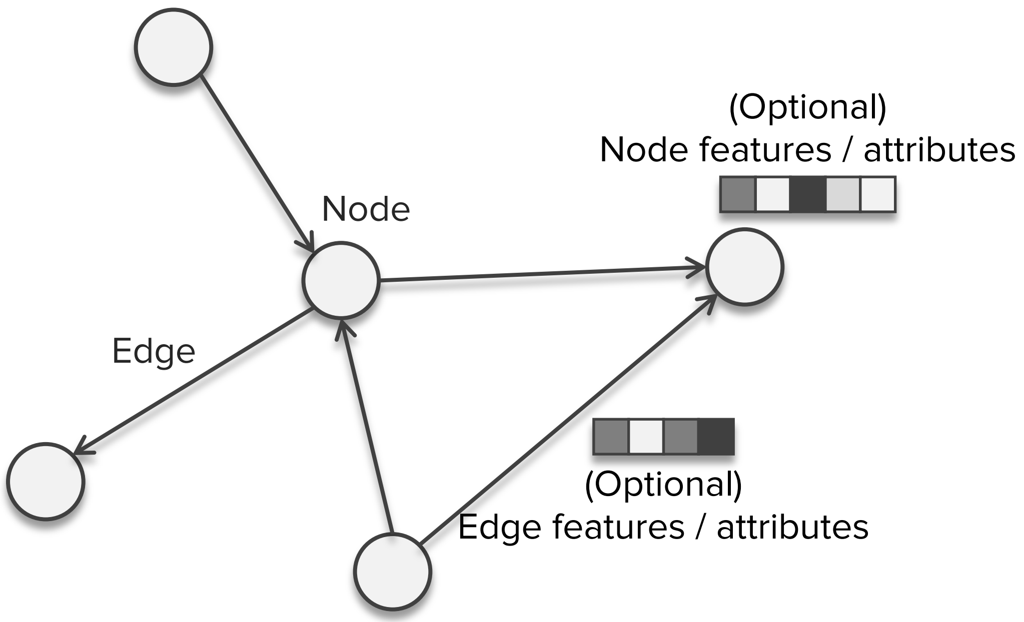
Understanding the general architecture of graph neural networks

Graphs as an abstract datatype

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



Undirected graph



Directed graph

...But honestly, looking at this does not result in a **practical** understanding of graphs.

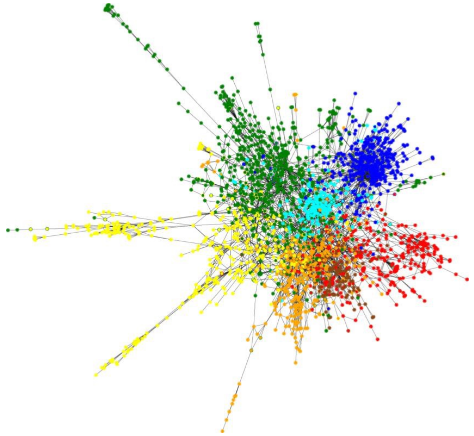
Therefore, we will look at **various benchmark datasets** in the field of **graph machine learning**.

In academia: Benchmark datasets in the literature

Social

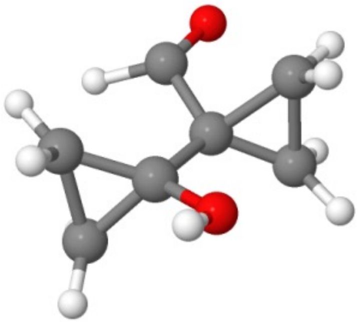
Node: People / Account
Edge: Connection
Node feature: Metadata

Citation / Web



Node: Paper
Edge: Citation
Node feature: Abstract

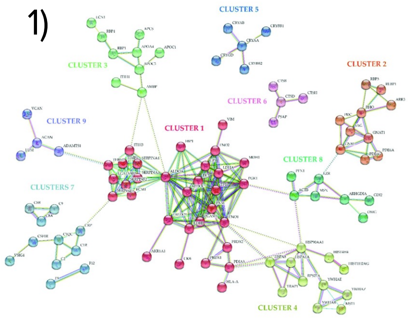
Molecules



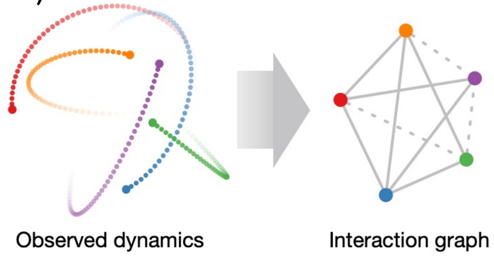
Node: Atom
Edge: Bond
Node feature: Atom type
Edge feature: Bond type

Biology / Simulation / etc.

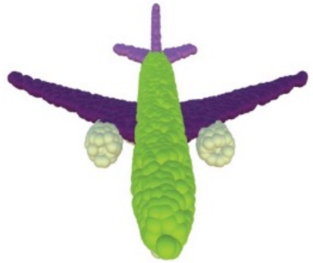
1)



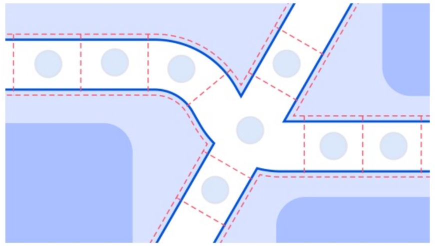
2)



3)



4)



Example benchmark datasets

- *Planetoid dataset (Cora/Citeseer/Pubmed)
- Coauthor
- WebKB (Texas/Cornell/etc.)

- Reddit
- Ego-Facebook
- Github

- QM9
- Zinc
- MUTAG

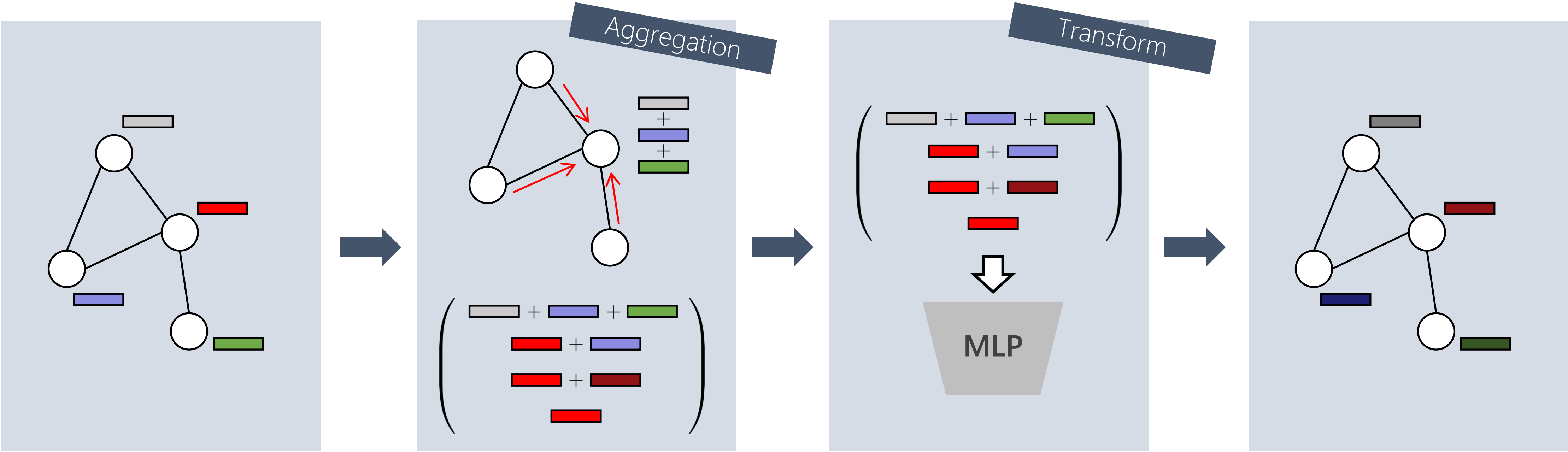
...

1) **PPI (protein-protein interaction)
2) Physical simulation (Kipf et al., 2018)
3) 3D point cloud (Wang et al., 2019)
4) Road network (Derrow-Pinion et al., 2021)

Yang et al., Revisiting Semi-Supervised Learning with Graph Embeddings, ICML 2016
Kipf et al., Neural Relational Inference for Interacting Systems, ICML 2018
Wang et al., Dynamic Graph CNN for Learning on Point Clouds, ACM Transactions on Graphics 2019
Derrow-Pinion et al., ETA Prediction with Graph Neural Networks in Google Maps, CIKM 2021
**Image source: https://www.researchgate.net/publication/324457787_iTRAQ_Quantitative_Proteomic_Analysis_of_Vitreous_from_Patients_with_Retinal_Detachment/figures?lo=1

An illustration of a generic GNN layer's operation

This is how a typical single layer of GNN operates when it calculates node representations/embedding vectors.



Therefore, the GNN can encode **node feature vectors** (if exists) + **edge feature vectors** (if exists) + **graph structure** (directly determines which vectors to aggregate).

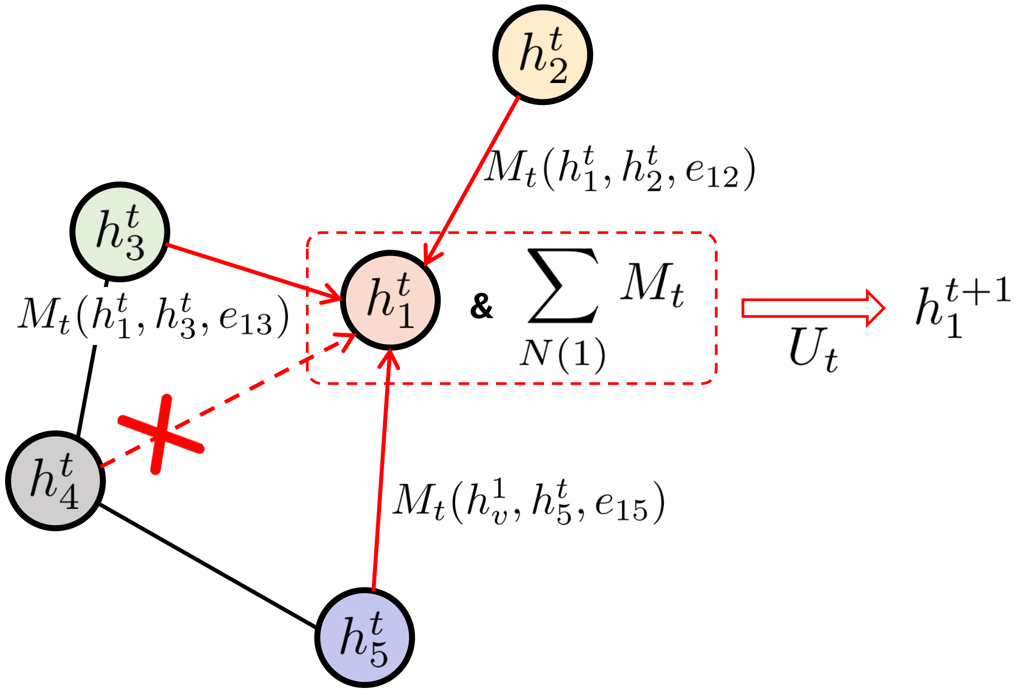
Abstraction: A general message-passing layer of GNNs

1. Message passing phase (Aggregation)

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

2. Update phase (Transformation)

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

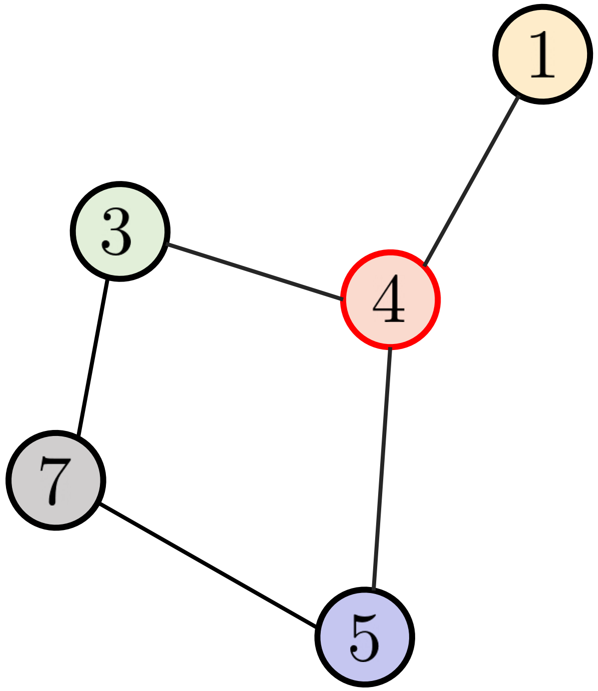


Note that this is simply a rewriting of the same concept from the previous slide.

*Usually, we cite these papers for the term “message-passing”
[First formal introduction of the concept] Gilmer et al., “Neural Message Passing for Quantum Chemistry”, ICML 2017
[Comprehensive discussion & abstraction] Bronstein et al., Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, arXiv 2021

Abstraction: A general message-passing layer of GNNs

GNN layer (Message-passing neural networks)



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

$\bigoplus$

This operation must be permutation invariant to ensure the same result for arbitrary node orderings!
Summation / Average / (Max) pooling etc.

So if we re-describe GCN (Graph convolutional network) for node 4, it would be...

$$\mathcal{N}_u = \{1, 3, 5\} \cup \{4\} \quad \psi(\mathbf{x}_u, \mathbf{x}_1) = \frac{1}{\sqrt{2 \times 4}}\mathbf{x}_1 \quad \phi = \mathrm{MLP}$$

*Usually, we cite these papers for the term "message-passing"
[First formal introduction of the concept] Gilmer et al., "Neural Message Passing for Quantum Chemistry", ICML 2017
[Comprehensive discussion & abstraction] Bronstein et al., Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, arXiv 2021

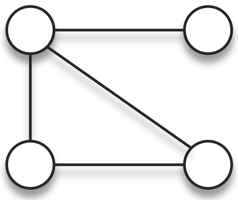
On representational power in GNNs

What makes two graphs the 'same'?

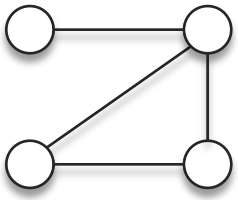
What do we mean by representational power?

Basically, GNNs have 'good representational power' if
they can tell two different graphs apart & perceive same graphs identically.

So how do we define if two graphs are different?



vs.

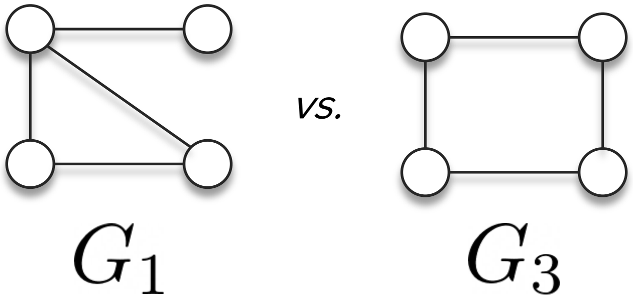


G_1 vs. G_2

What do we mean by representational power?

Basically, GNNs have ‘good representational power’ if
they can tell two different graphs apart & perceive same graphs identically.

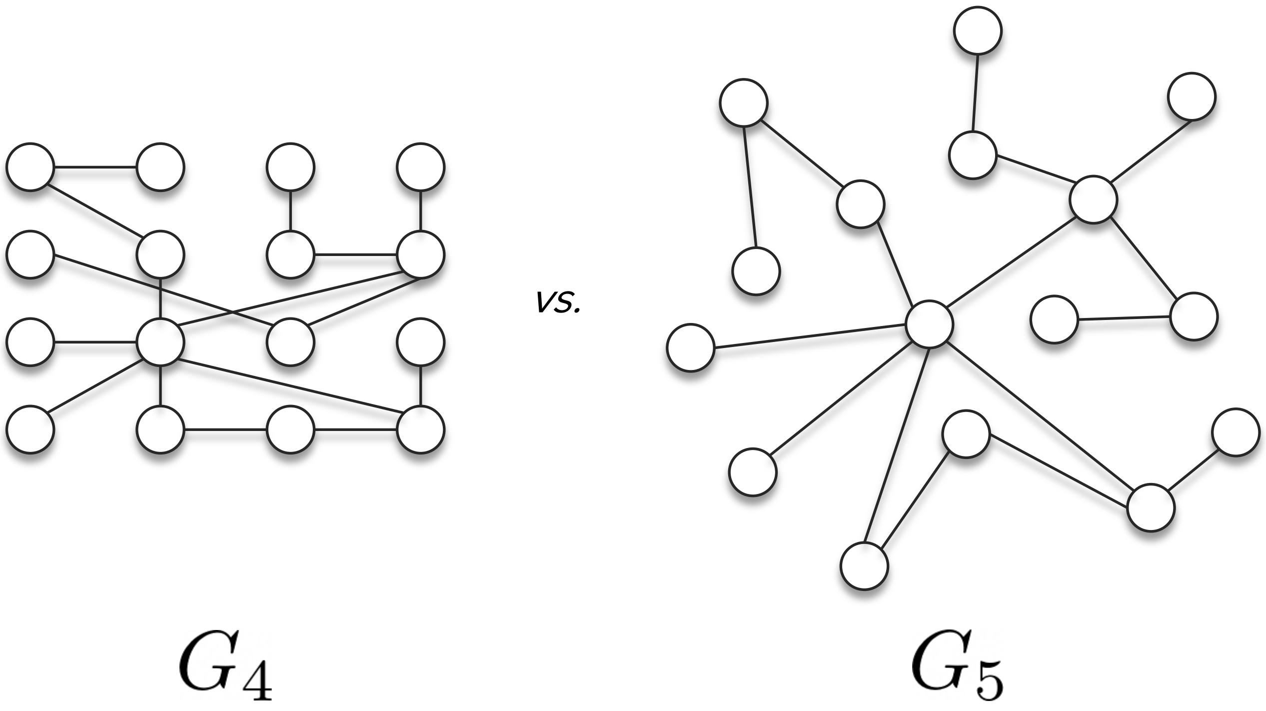
So how do we define if two graphs are different?



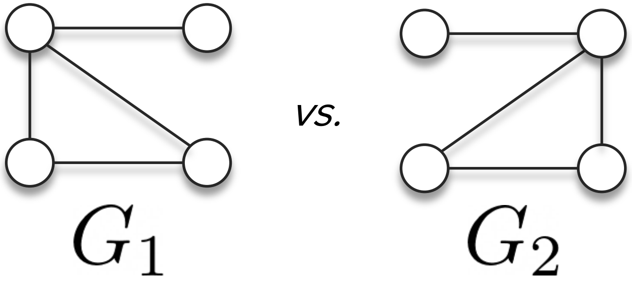
What do we mean by representational power?

Basically, GNNs have 'good representational power' if
they can tell two different graphs apart & perceive same graphs identically.

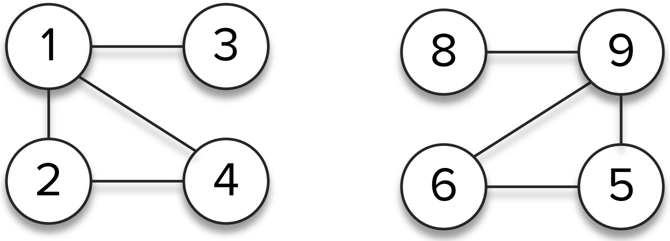
So how do we define if two graphs are different?



Isomorphism (a fancy word for identical graphs)



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



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

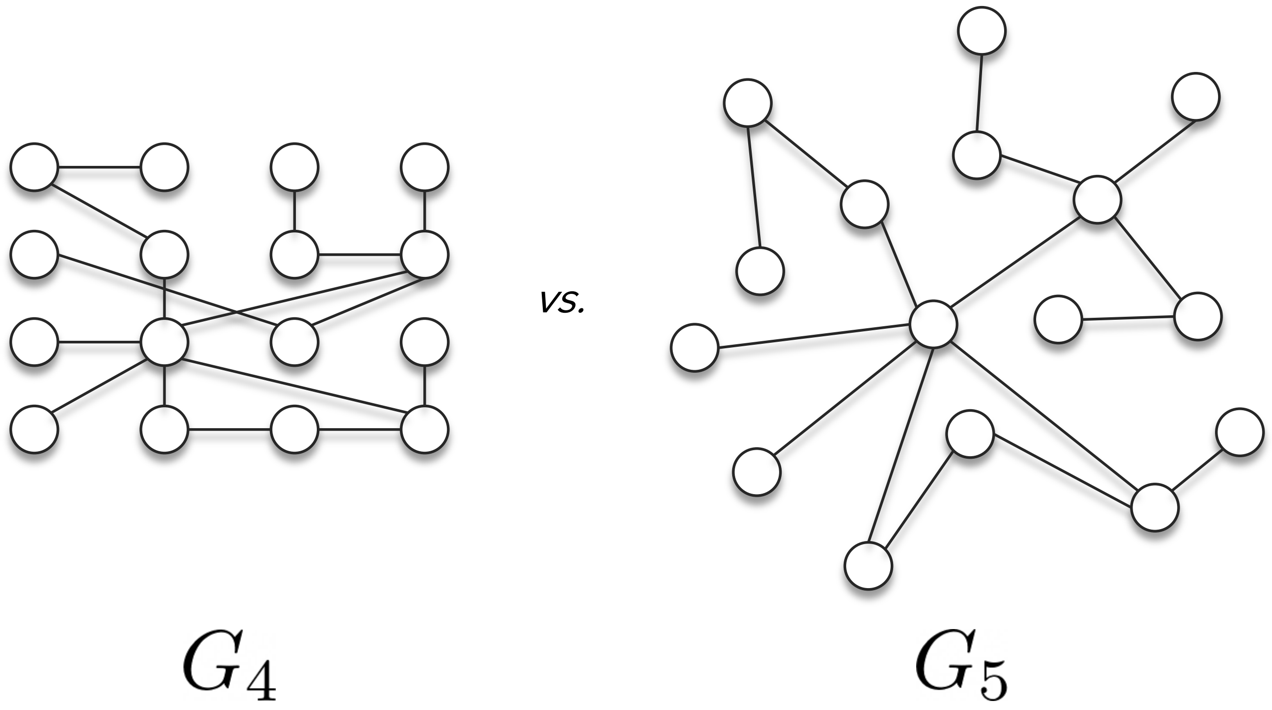
This means, G1 and G2 are **isomorphic** since we **can** find a bijection mapping of:

$3 \leftrightarrow 8$
$1 \leftrightarrow 9$
$4 \leftrightarrow 6$
$2 \leftrightarrow 5$

and according to this node mapping, the edge set from G1 exactly translates to G2 and therefore **G1 is the 'same' graph as G2.**

[1] (Definition) Morris et al., Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks, AAAI 2019

The practical problem of graph isomorphism test



*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

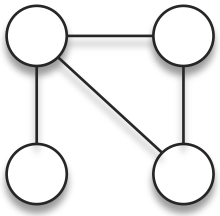
[2] Huang & Villar, "A short tutorial on the Weisfeiler-Lehman test and its variants", ICASSP 2021
[3] David Bieber, "The Weisfeiler-Lehman Isomorphism Test" (Blog post)

Method for testing between two graphs

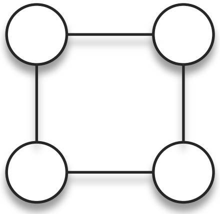
Understanding the WL-isomorphism test

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

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



Graph 1

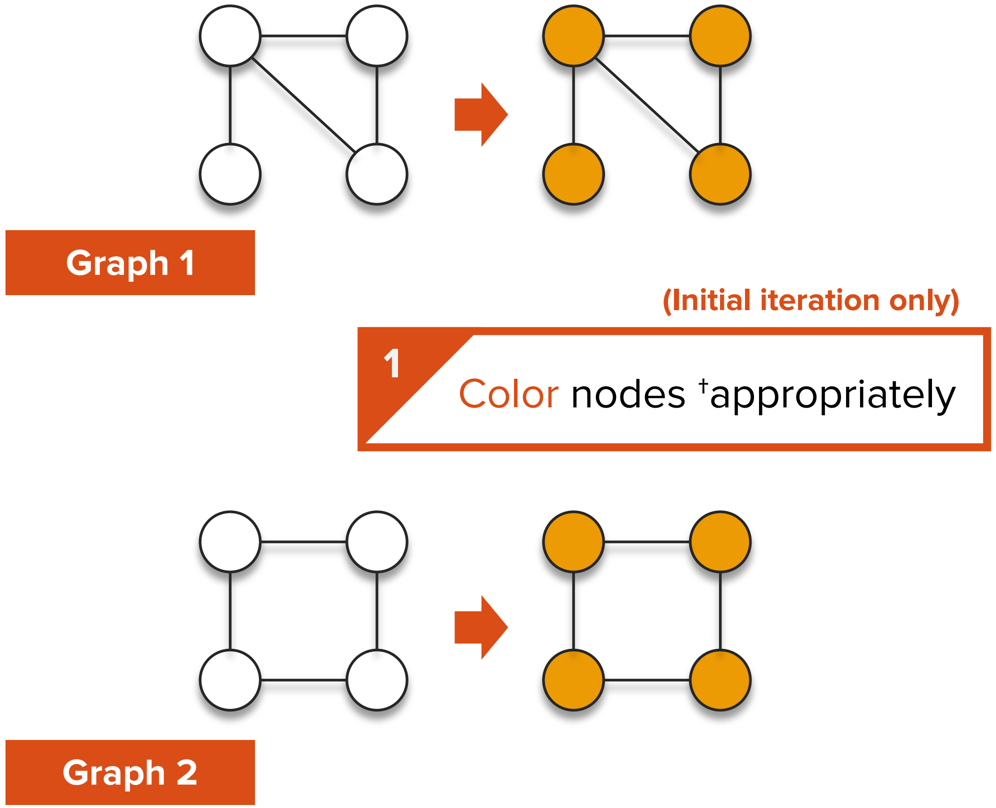


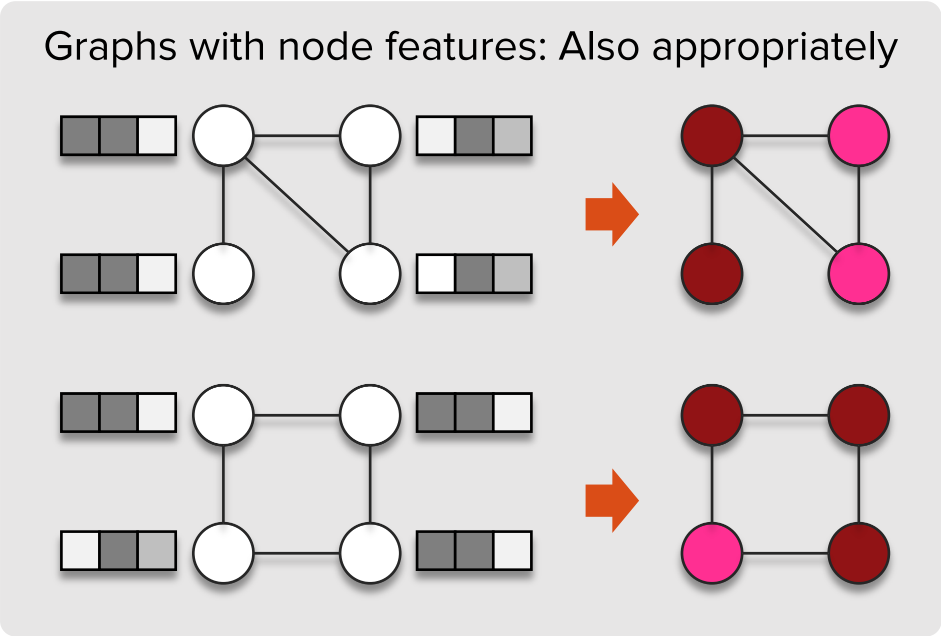
Graph 2

†Specifically, we are showing the simplest version of the WL test, which is also known as the color refinement algorithm (for reasons which will be apparent momentarily)

[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]

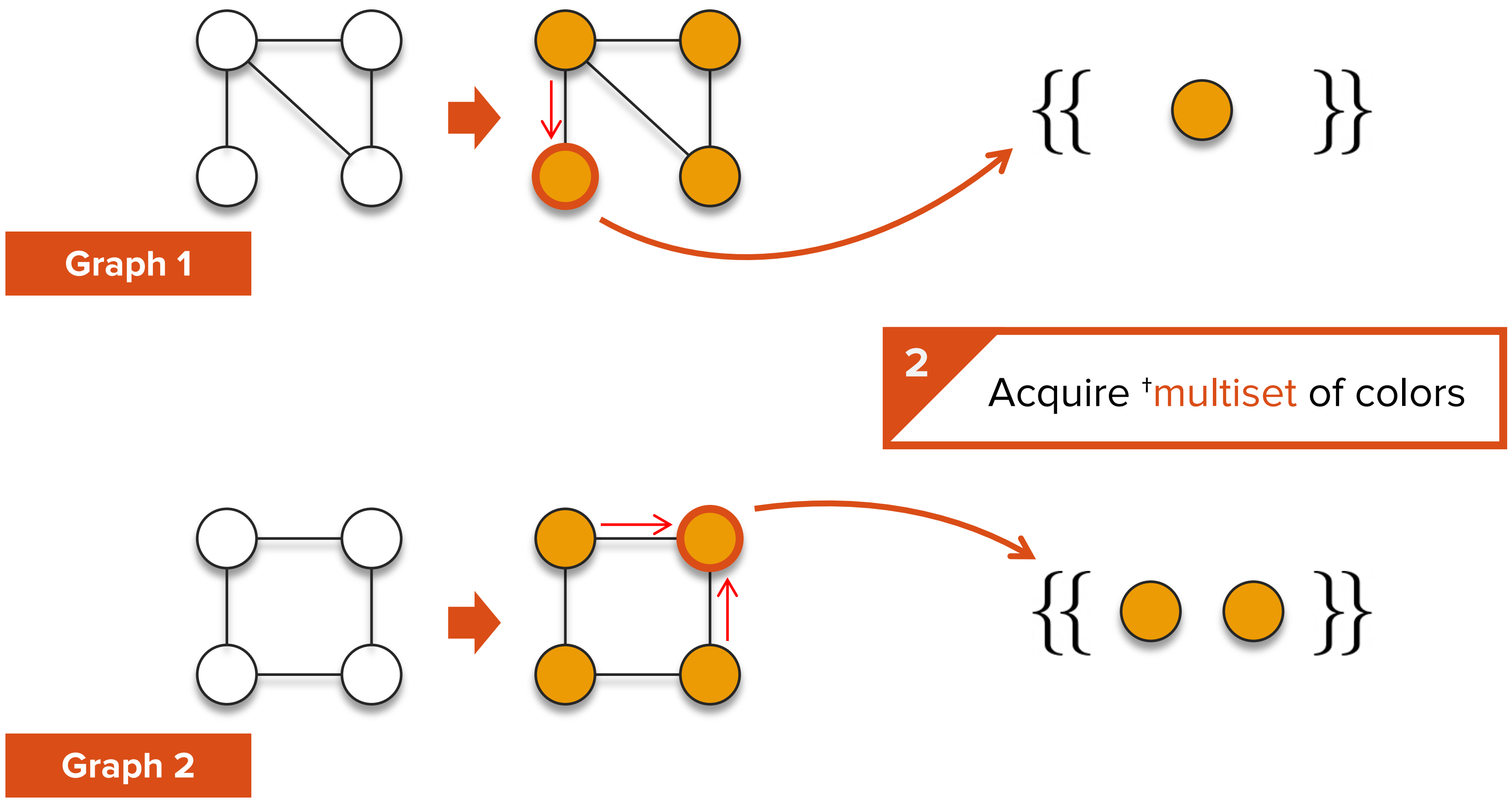




†As suggested by [4], color node according to the node degree. Or just start with a uniform coloring

[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)

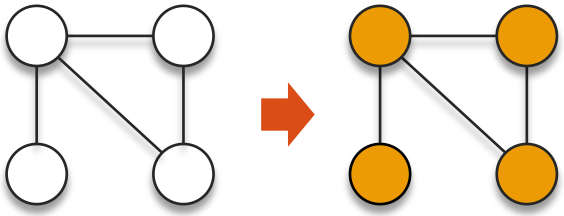
One iteration of the WL-isomorphism test [1], [2]

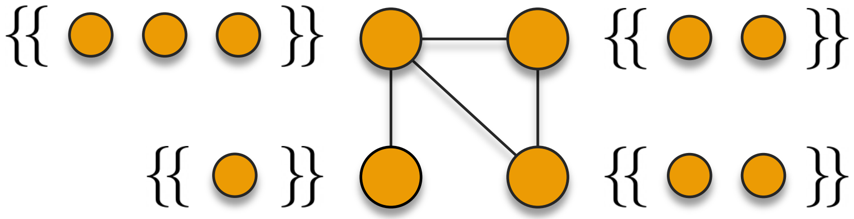


†Multiset is a set that allows multiple duplicates of elements

[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)

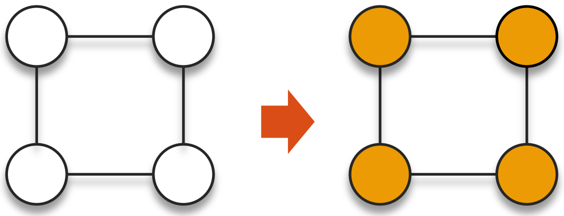
One iteration of the WL-isomorphism test [1], [2]

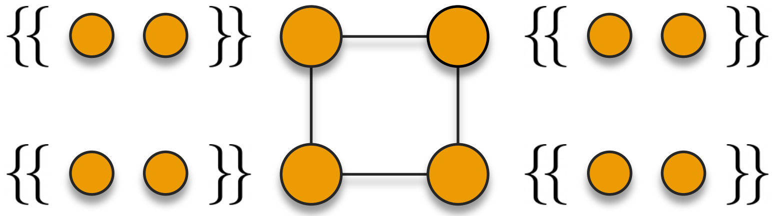




Graph 1

2 Acquire †multiset of colors



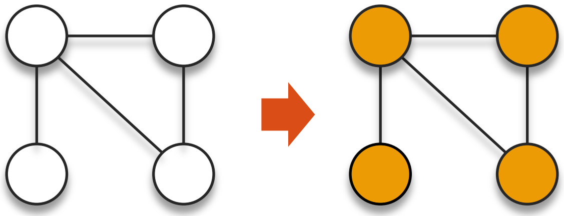


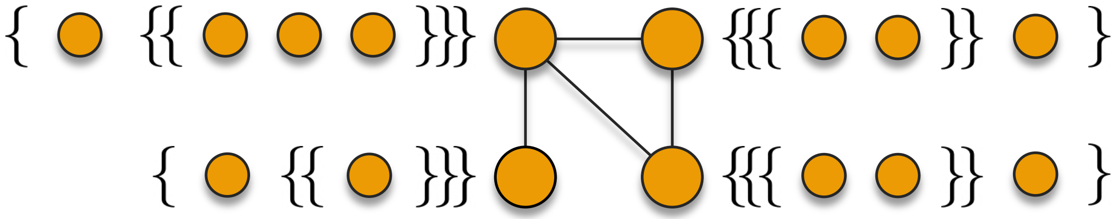
Graph 2

†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)

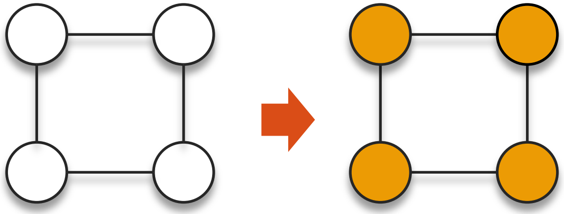
One iteration of the WL-isomorphism test [1], [2]

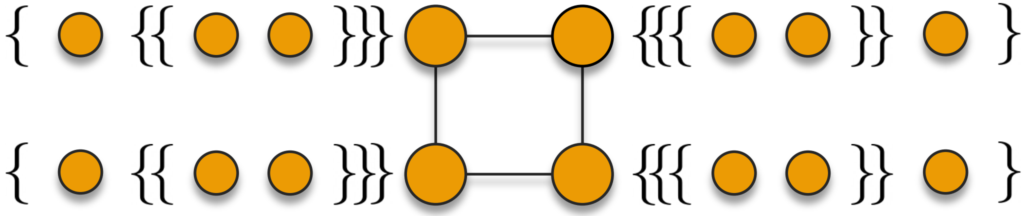




Graph 1

3 Make a set by including self



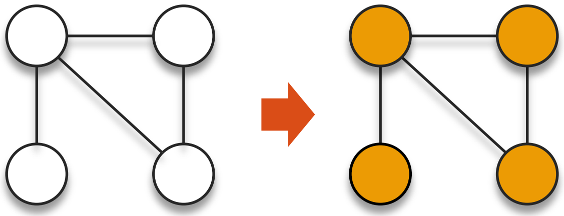


Graph 2

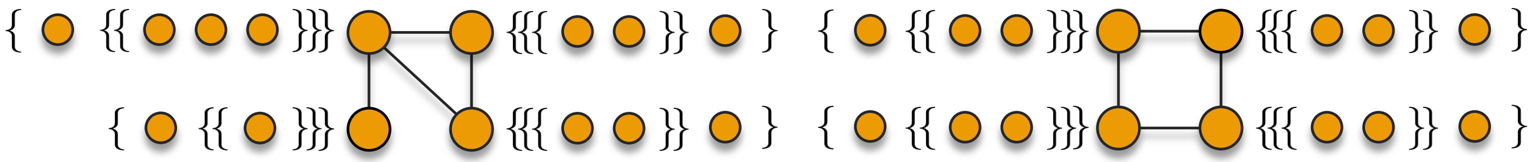
†Multiset is a set that allows multiple duplicates of elements

[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)

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

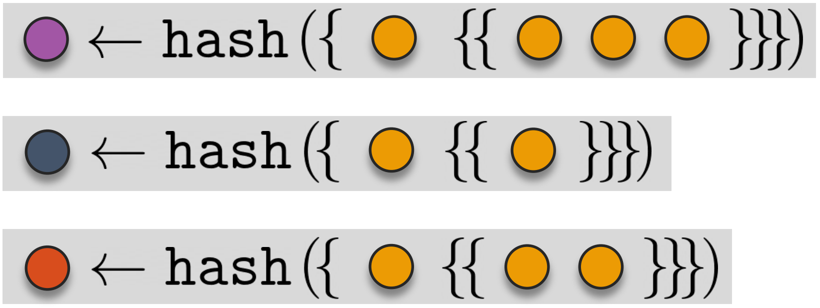


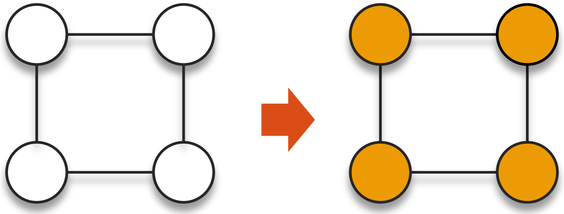
Graph 1



4

Map each set to a new color by a †bijective function



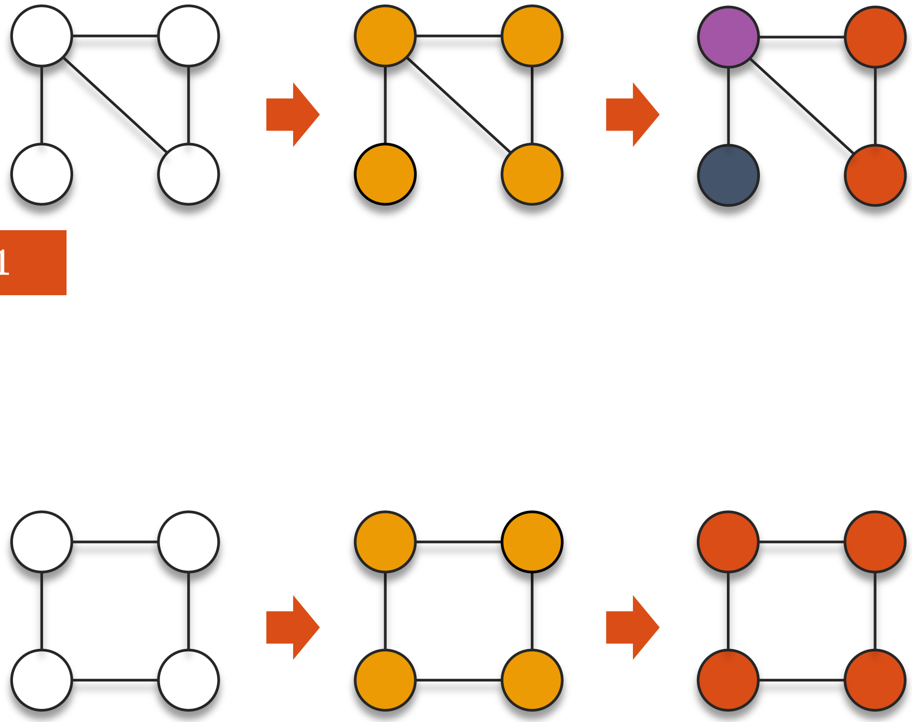


Graph 2

† 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)

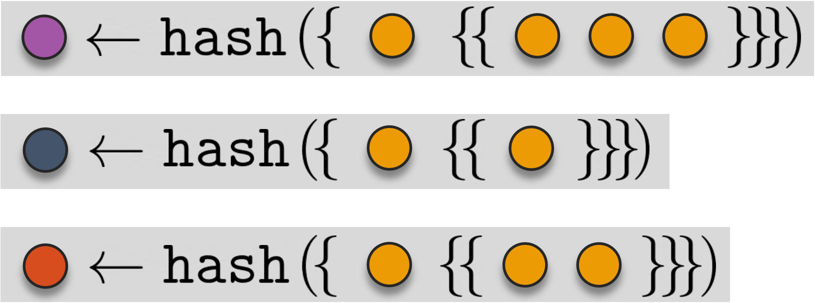
One iteration of the WL-isomorphism test [1], [2]



Graph 1

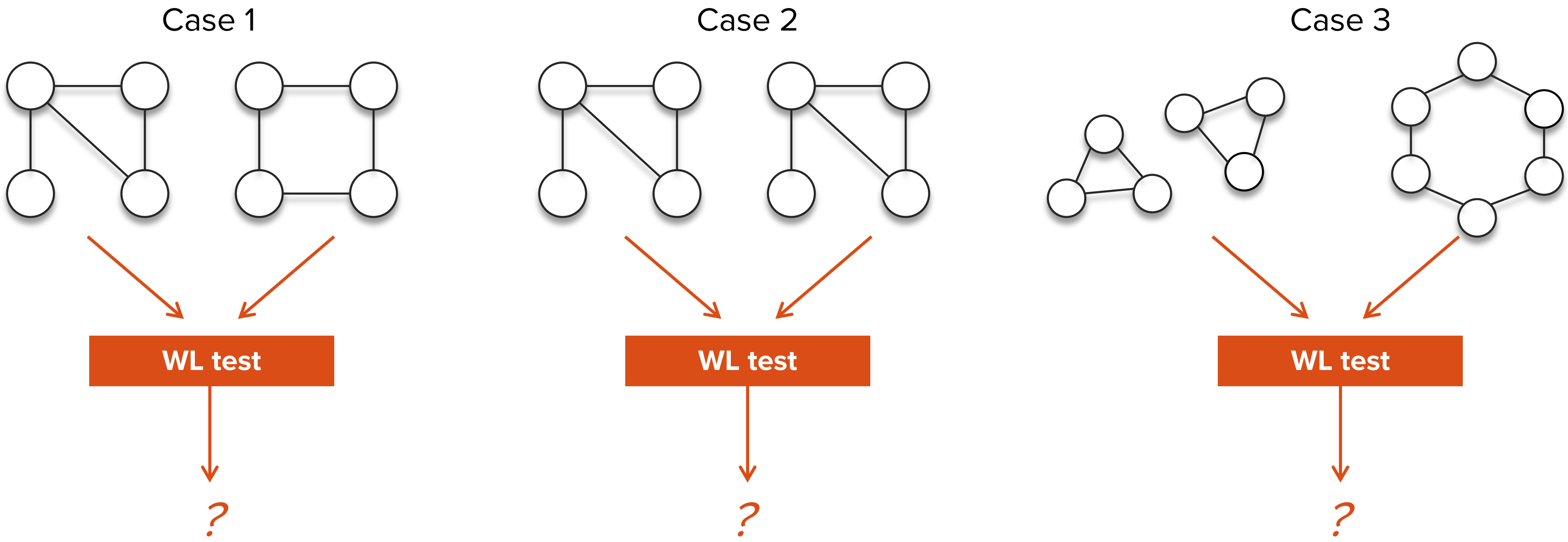
Graph 2

5 Get the colors of the next iteration



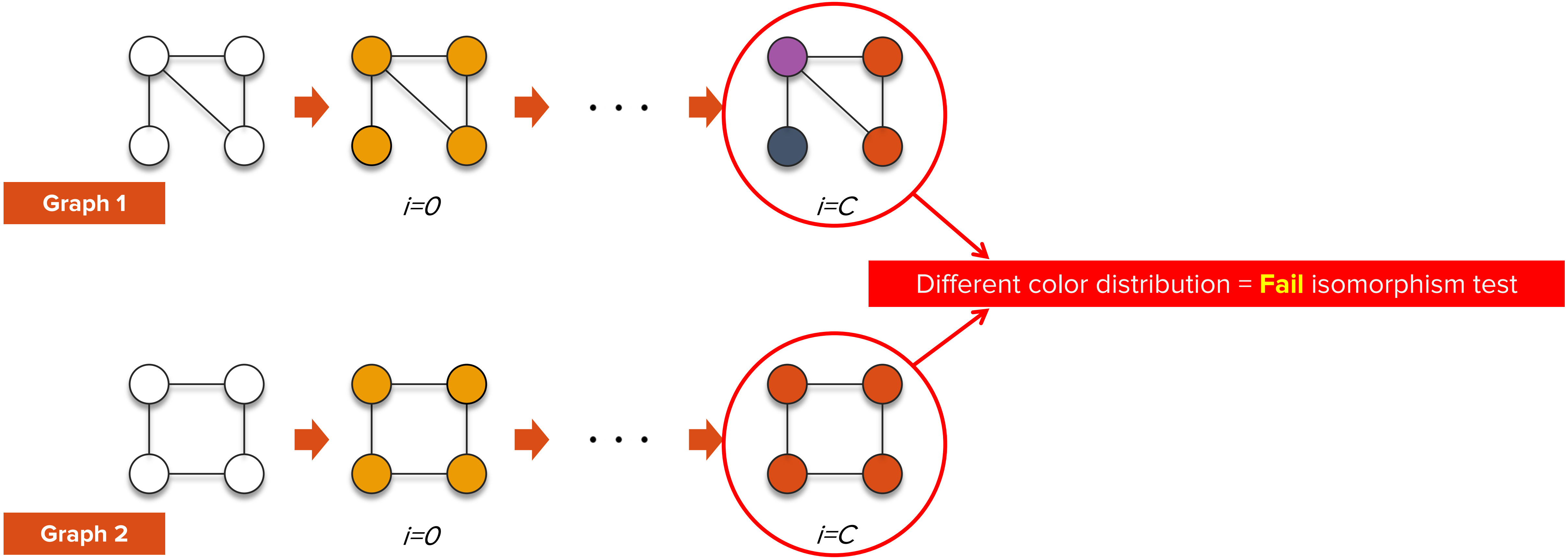
[4] Shervashidze et al., "Weisfeiler-Lehman Graph Kernels", J. Mach. Learn. Res. (2011)
[5] Morris et al,. "Weisfeiler and Leman go Machine Learning: The Story so far", arXiv (2021)

WL-isomorphism test: Three example cases



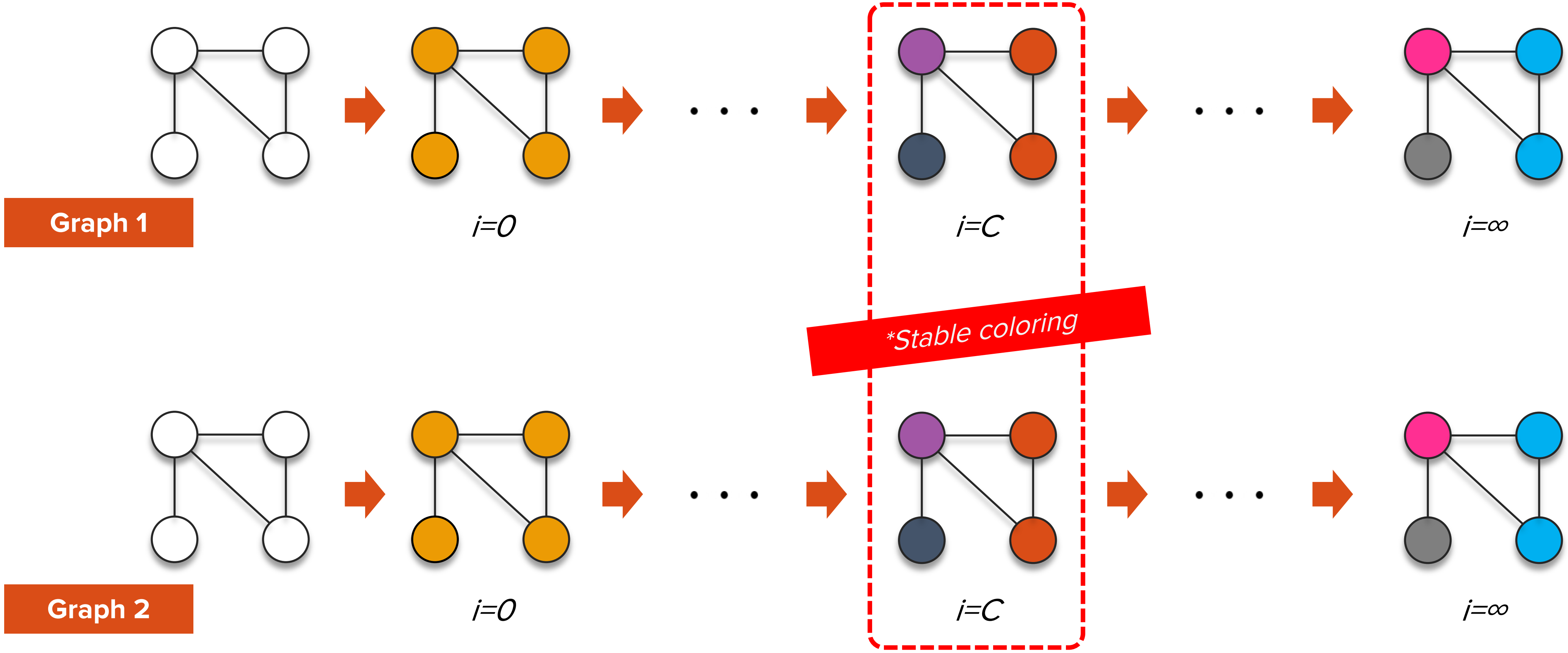
WL-isomorphism test: Three example cases

Counclusion of case 1



WL-isomorphism test: Three example cases

Counclusion of case 2

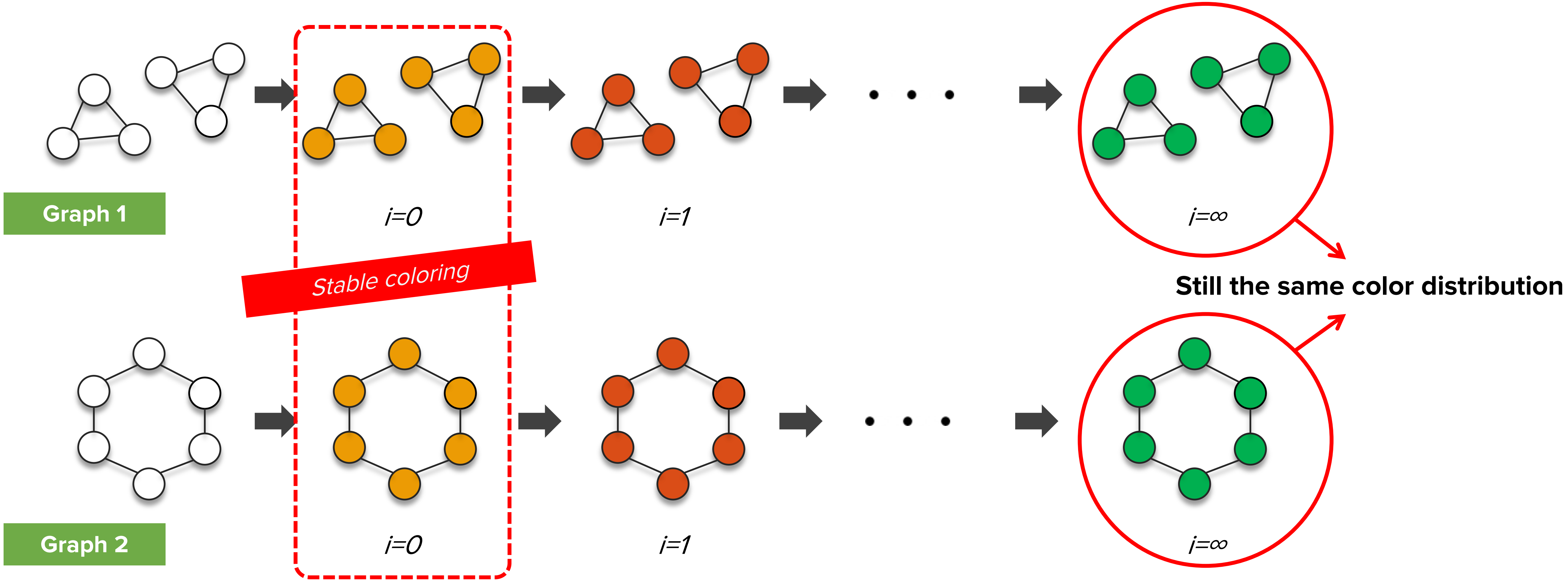


Conclusion: Two graphs are isomorphic ..?

* 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*(|Graph 1|, |Graph 2|) (see [5]).

WL-isomorphism test: Three example cases

Counclusion of case 3

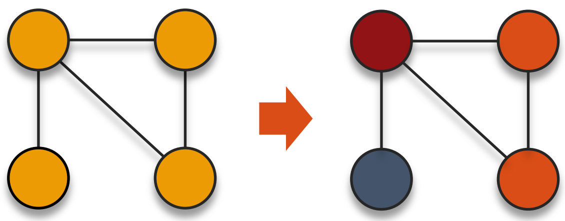


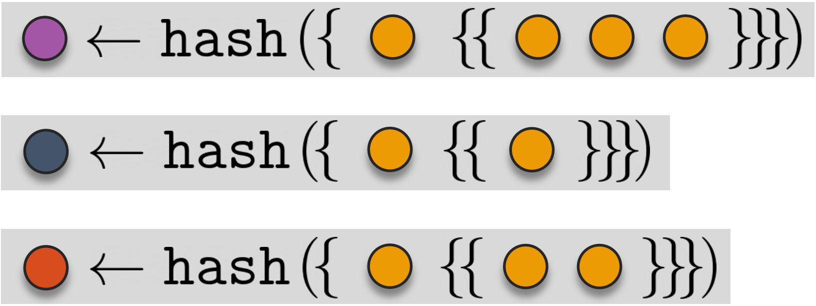
Conclusion: Two graphs are isomorphic ✗ **Cannot determine** ✓

Understanding the connection between the WL test and message-passing

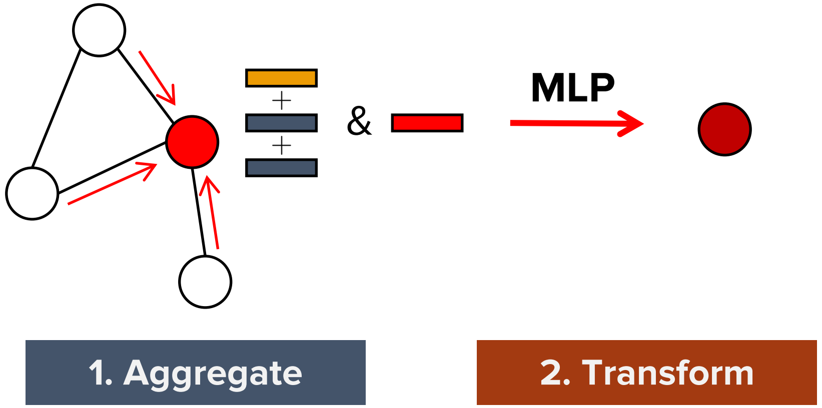
Relation between WL and GNNs

“Color refinement” in WL





Message passing in GNNs

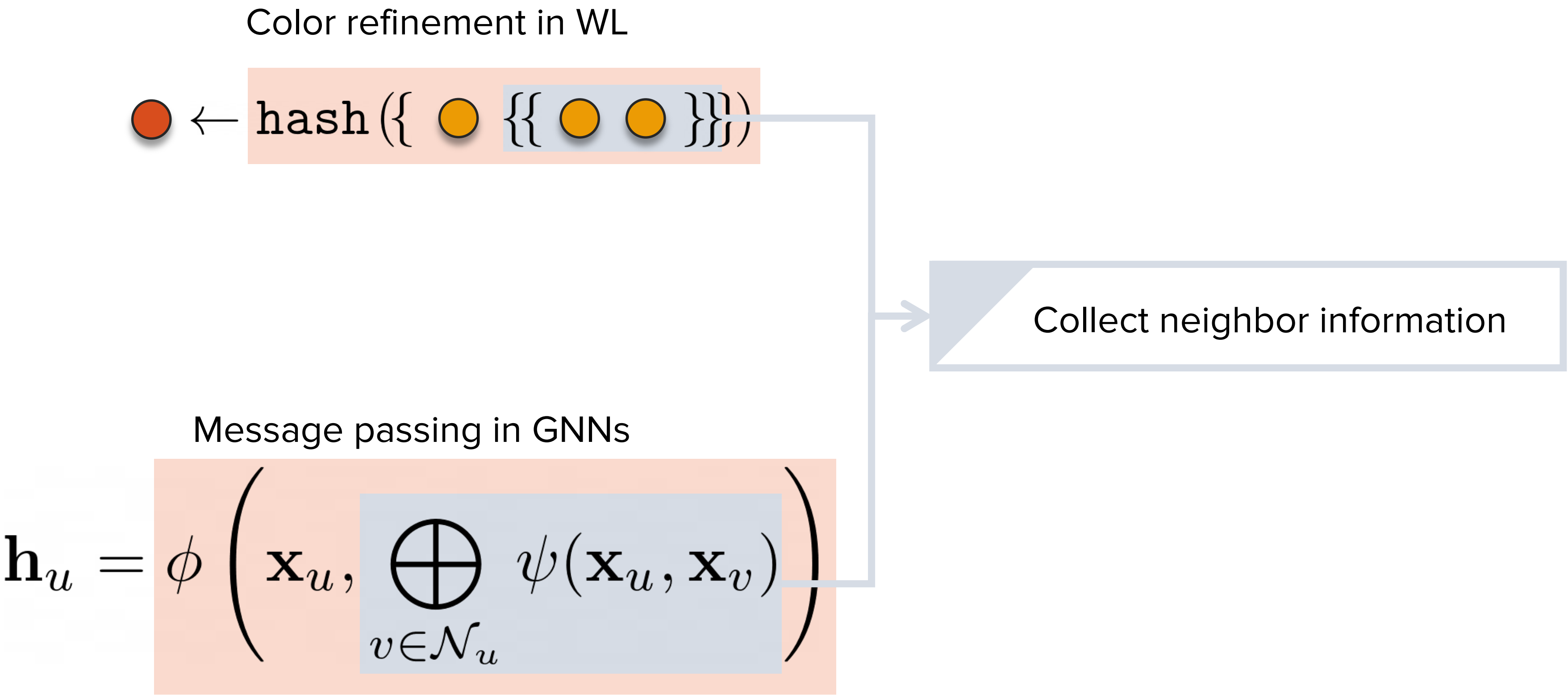


$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

Can you see the similarity?

Relation between WL and GNNs

From a feed forward computational standpoint, GNNs are a neural network version of the WL test.

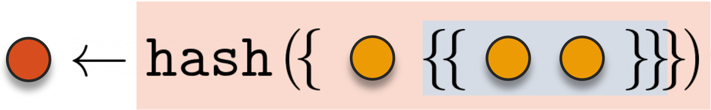


$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right)$$

Relation between WL and GNNs

From a feed forward computational standpoint, GNNs are a neural network version of the WL test.

Color refinement in WL



Map self & neighbor information to next iteration

Message passing in GNNs

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

Consequences of GNN's ability to differentiate graphs

Color refinement in WL

$$\bullet \leftarrow \texttt{hash}(\{\bullet\ \{\{\bullet\ \bullet\}\}\})$$

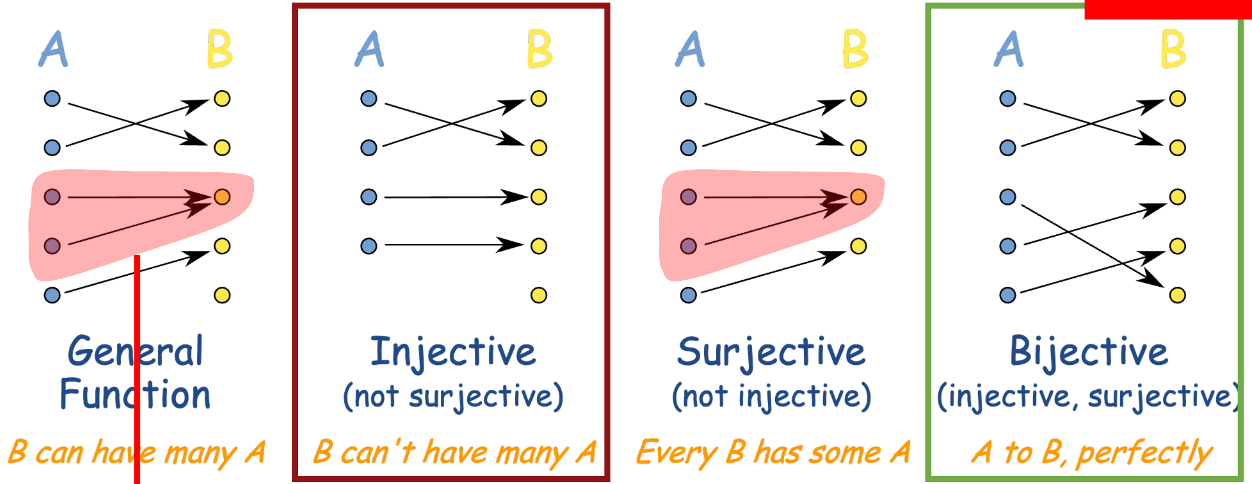
- $\texttt{hash}$: Fixed **bijective** function (at least **injective**)

Message passing in GNNs

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

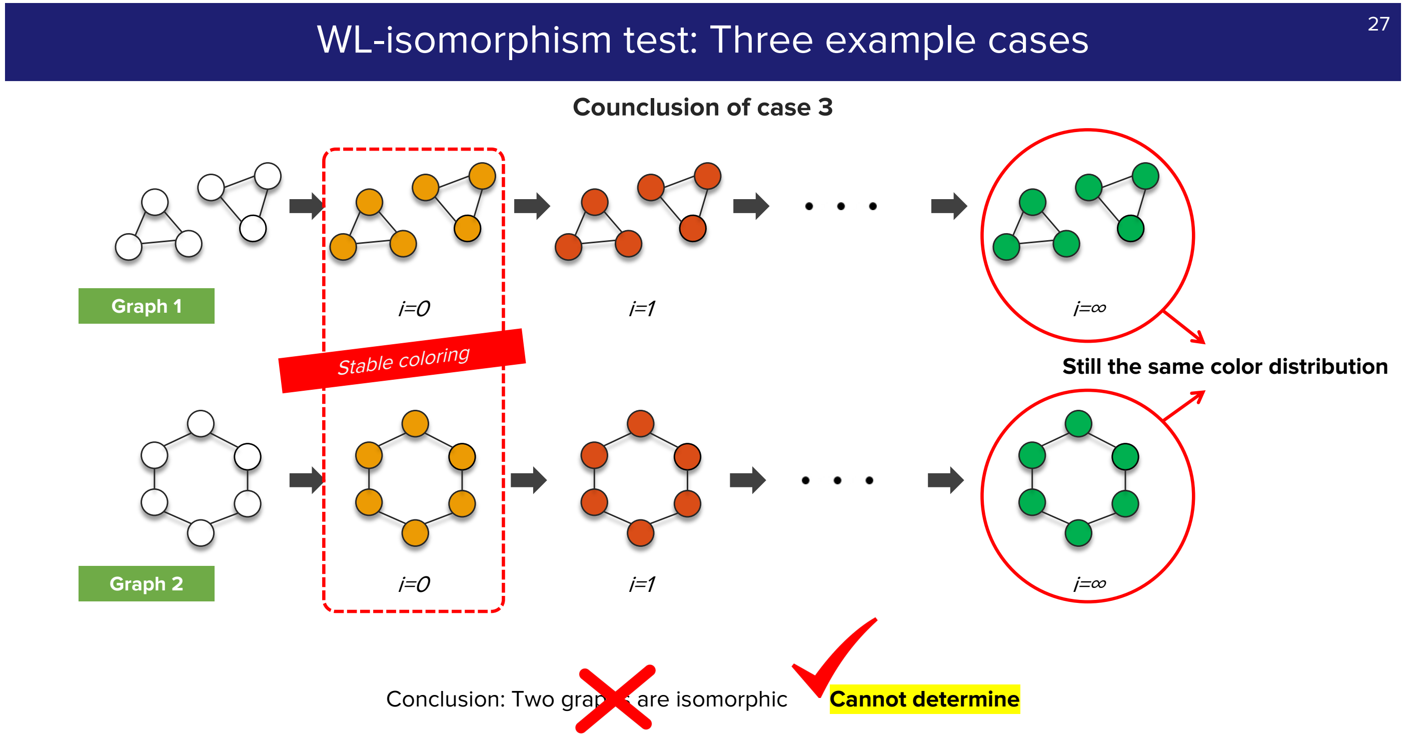
- ϕ, ψ: A neural network (Learnalbe weights)
- (Probably) *Not* **bijective** *nor* **injective**

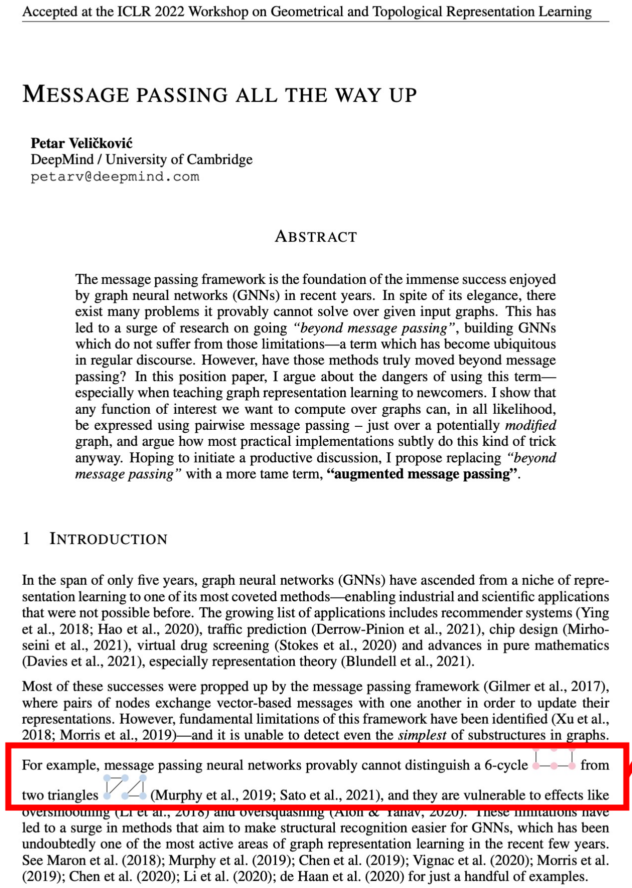
GNNs are *at best* 1-WL



Loss of expressive power: Cannot distinguish some elements

Consequences of GNN's ability to differentiate graphs




Accepted at the ICLR 2022 Workshop on Geometrical and Topological Representation Learning

MESSAGE PASSING ALL THE WAY UP

Petar Veličković
DeepMind / University of Cambridge
petarv@deepmind.com

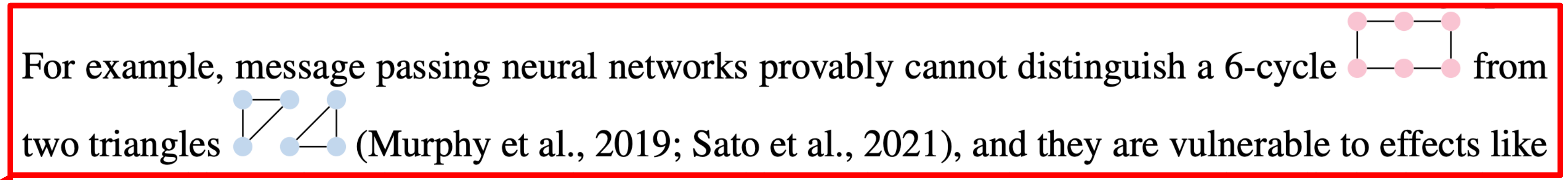
ABSTRACT

The message passing framework is the foundation of the immense success enjoyed by graph neural networks (GNNs) in recent years. In spite of its elegance, there exist many problems it provably cannot solve over given input graphs. This has led to a surge of research on going *"beyond message passing"*, building GNNs which do not suffer from those limitations—a term which has become ubiquitous in regular discourse. However, have those methods truly moved beyond message passing? In this position paper, I argue about the dangers of using this term—especially when teaching graph representation learning to newcomers. I show that any function of interest we want to compute over graphs can, in all likelihood, be expressed using pairwise message passing – just over a potentially *modified* graph, and argue how most practical implementations subtly do this kind of trick anyway. Hoping to initiate a productive discussion, I propose replacing *"beyond message passing"* with a more tame term, **"augmented message passing"**.

1 INTRODUCTION

In the span of only five years, graph neural networks (GNNs) have ascended from a niche of representation learning to one of its most coveted methods—enabling industrial and scientific applications that were not possible before. The growing list of applications includes recommender systems (Ying et al., 2018; Hao et al., 2020), traffic prediction (Derrow-Pinion et al., 2021), chip design (Mirhoseini et al., 2021), virtual drug screening (Stokes et al., 2020) and advances in pure mathematics (Davies et al., 2021), especially representation theory (Blundell et al., 2021).

Most of these successes were propped up by the message passing framework (Gilmer et al., 2017), where pairs of nodes exchange vector-based messages with one another in order to update their representations. However, fundamental limitations of this framework have been identified (Xu et al., 2018; Morris et al., 2019)—and it is unable to detect even the *simplest* of substructures in graphs. For example, message passing neural networks provably cannot distinguish a 6-cycle from two triangles (Murphy et al., 2019; Sato et al., 2021), and they are vulnerable to effects like oversmoothing (Li et al., 2018) and oversquashing (Alon & Yahav, 2020). These limitations have led to a surge in methods that aim to make structural recognition easier for GNNs, which has been undoubtedly one of the most active areas of graph representation learning in the recent few years. See Maron et al. (2018); Murphy et al. (2019); Chen et al. (2019); Vignac et al. (2020); Morris et al. (2019); Chen et al. (2020); Li et al. (2020); de Haan et al. (2020) for just a handful of examples.


For example, message passing neural networks provably cannot distinguish a 6-cycle from two triangles (Murphy et al., 2019; Sato et al., 2021), and they are vulnerable to effects like

And therefore GNNs also inherit the same limitations from WL.

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

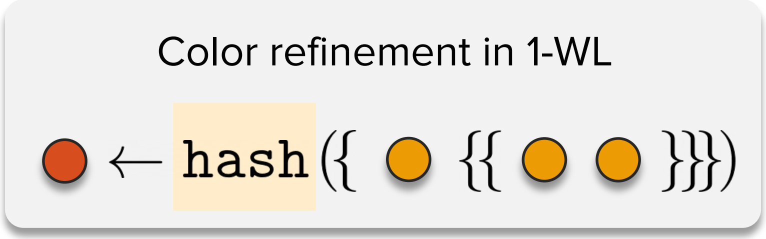
GNNs cannot exceed WL in terms of its expressivity

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.



$\geq$

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?

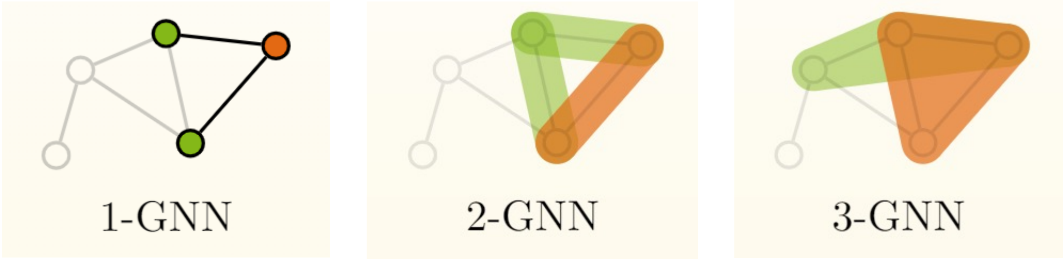
Problem: GNNs are bound by 1-dim WL-test

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

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

Hence, in the light of the above results, 1-GNNs may 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.

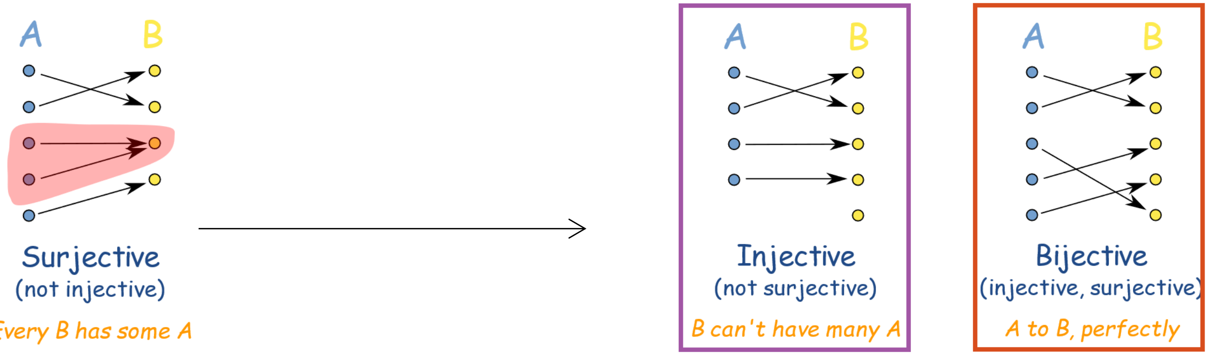


GNNs cannot exceed WL in terms of its expressivity

Space of *general functions
(Expressed by neural networks)

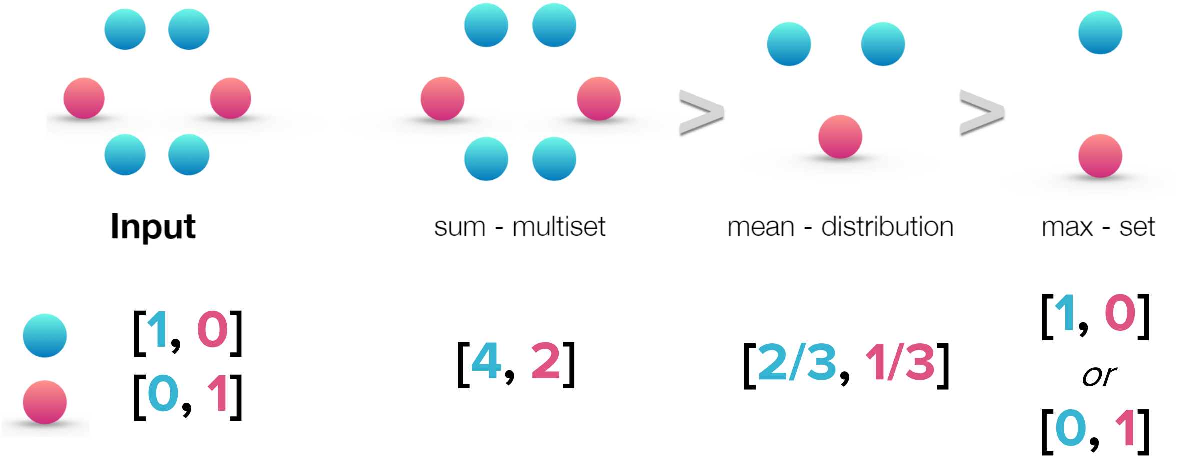
Most functions

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



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

1. Use summation for aggregation



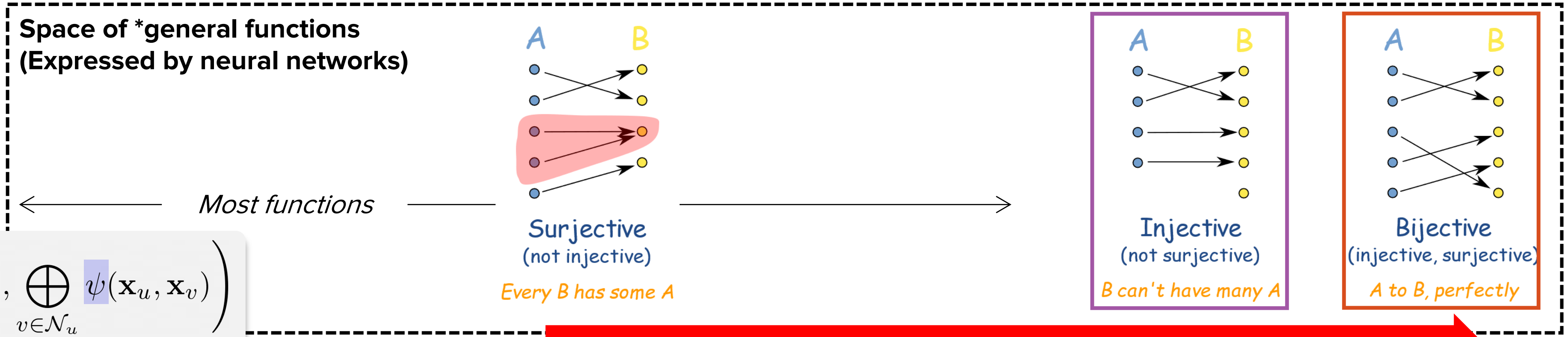
2. Use at least 2 layers of MLP

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



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

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

Graph Isomorphism Networks (GIN)

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

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

Takeaways

1. Defining graphs being ‘identical’ = isomorphism
2. WL-isomorphism test: Heuristic that can be used for isomorphism, but not guaranteed in some cases
3. Connections: GNN’s message-passing and WL test, and GNN’s limitations

Thank you!

Please feel free to ask any questions :)

yongmin.shin@lgresearch.ai

Original presentation file

Feed free to contact me via email or teams etc. regarding

- GNN/graph learning
- Explainable AI (including mechanistic interpretability)
- Or any ML discussions