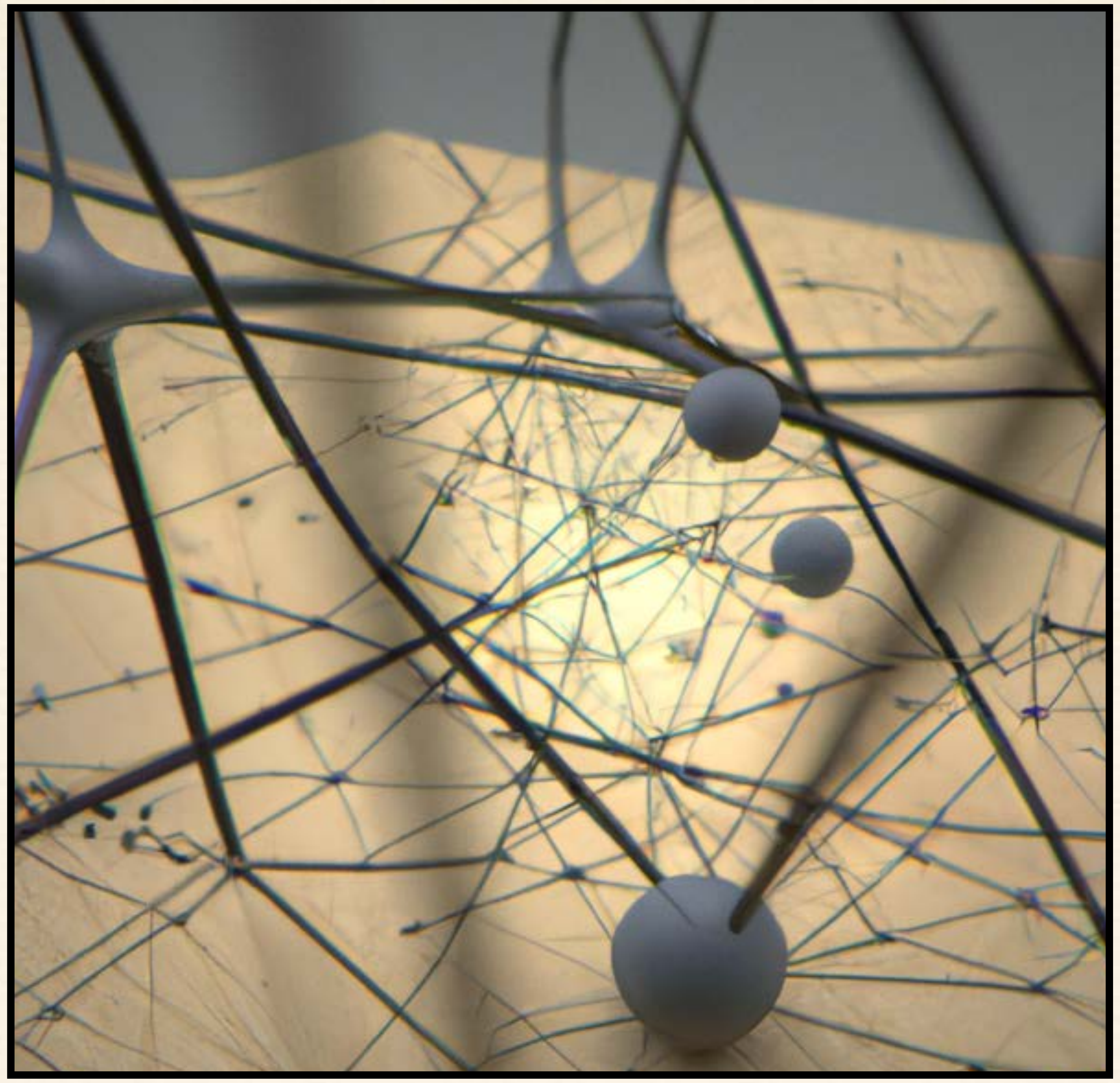


The (expressive) power of graph learning

Floris Geerts (University of Antwerp)

Course

- Is about recent advances in graph learning.
- With an emphasis on the expressive power of learning methods.
  - Self-contained (too some extent).
  - Mostly high-level, but also low-level, so basically all levels.
  - Not all methods or related works are covered.
  - Will not report experiments…

About the speaker

- Background in mathematics, database theory* and expressive power of query languages.
- Since 2018, expressive power of linear algebra.
- Natural move to the study of expressive power of graph neural networks.

2023

[c65] Floris Geerts:
A Query Language Perspective on Graph Learning. PODS 2023: 373-379

2022

[c64] Floris Geerts, Jasper Steegmans, Jan Van den Bussche:
On the Expressive Power of Message-Passing Neural Networks as Global Feature Map Transformers. FoIKS 2022: 20-34

[c63] Floris Geerts, Juan L. Reutter:
Expressiveness and Approximation Properties of Graph Neural Networks. ICLR 2022

[c62] Chendi Qian, Gaurav Rattan, Floris Geerts, Mathias Niepert, Christopher Morris:
Ordered Subgraph Aggregation Networks. NeurIPS 2022

2021

[c61] Floris Geerts, Filip Mazowiecki, Guillermo A. Pérez:
Let's Agree to Degree: Comparing Graph Convolutional Networks in the Message-Passing Framework. ICML 2021: 3640-3649

[c60] Pablo Barceló, Floris Geerts, Juan L. Reutter, Maksimilian Ryschkov:
Graph Neural Networks with Local Graph Parameters. NeurIPS 2021: 25280-25293

[c59] Floris Geerts, Thomas Muñoz, Cristian Riveros, Domagoj Vrgoc:
Expressive Power of Linear Algebra Query Languages. PODS 2021: 342-354

2020

[c58] Floris Geerts:
When Can Matrix Query Languages Discern Matrices? ICDT 2020: 12:1-12:18

[–] **2010 – 2019**

2019

[c57] Floris Geerts:
On the Expressive Power of Linear Algebra on Graphs. ICDT 2019: 7:1-7:19

[c56] Maarten Van den Heuvel, Peter Ivanov, Wolfgang Gatterbauer, Floris Geerts, Martin Theobald:
Anytime Approximation in Probabilistic Databases via Scaled Dissociations. SIGMOD Conference 2019: 1295-1312

2018

[c55] Robert Brijder, Floris Geerts, Jan Van den Bussche, Timmy Weerwag:
On the Expressive Power of Query Languages for Matrices. ICDT 2018: 10:1-10:17

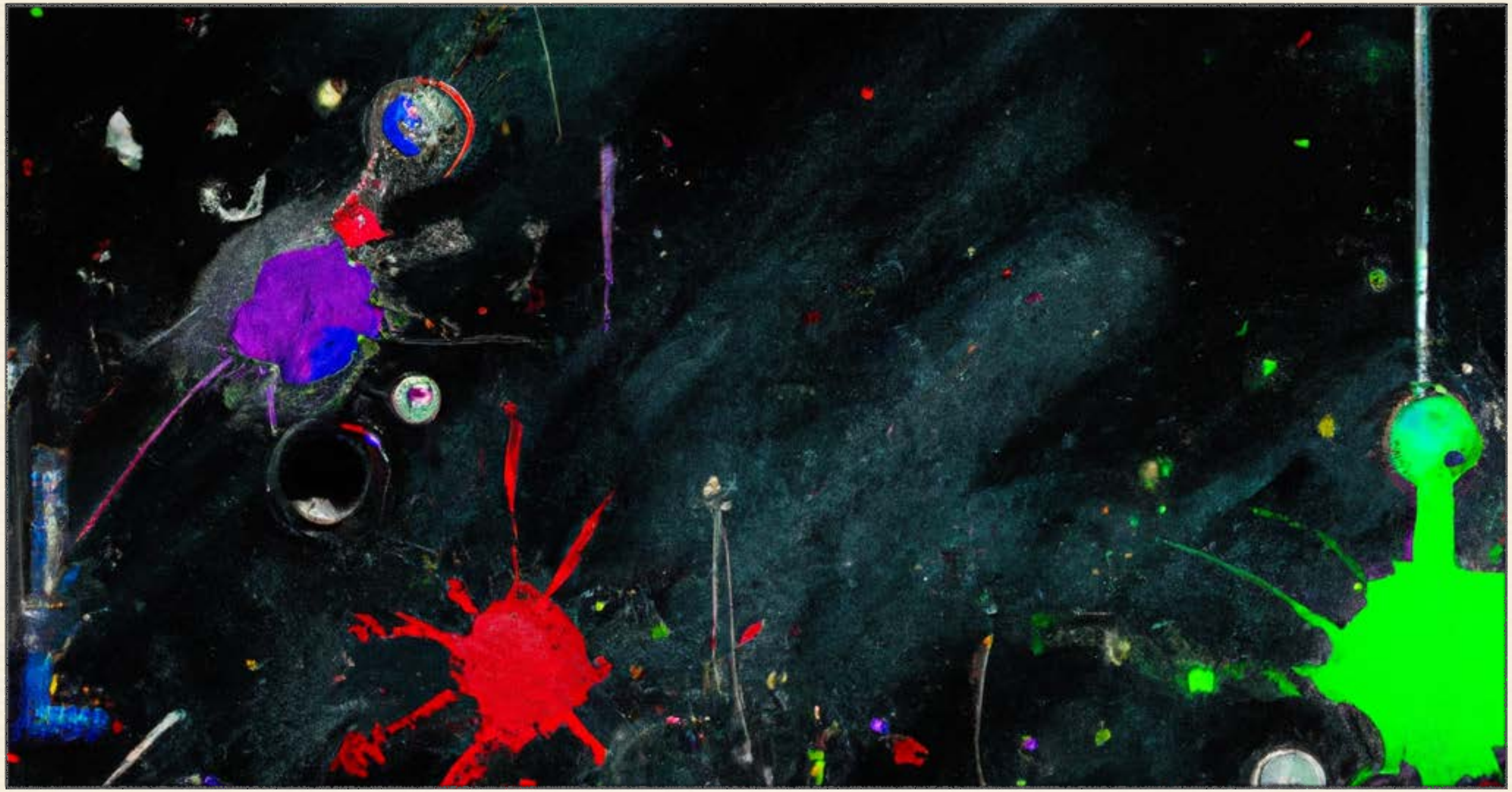
PhD in 2001. ;-(

Outline

- Graph learning and expressive power
- Message Passing Neural Networks
- Boosting power:
    - Feature augmentation
    - Subgraphs
    - Higher-order message-passing

Ask Questions

Indebted to Fabrizio Frasca, Beatrice Bevilacqua and Haggai Maron: Shamelessly :-) borrowed parts of their tutorial on expressive of GNNs at LOG 2022

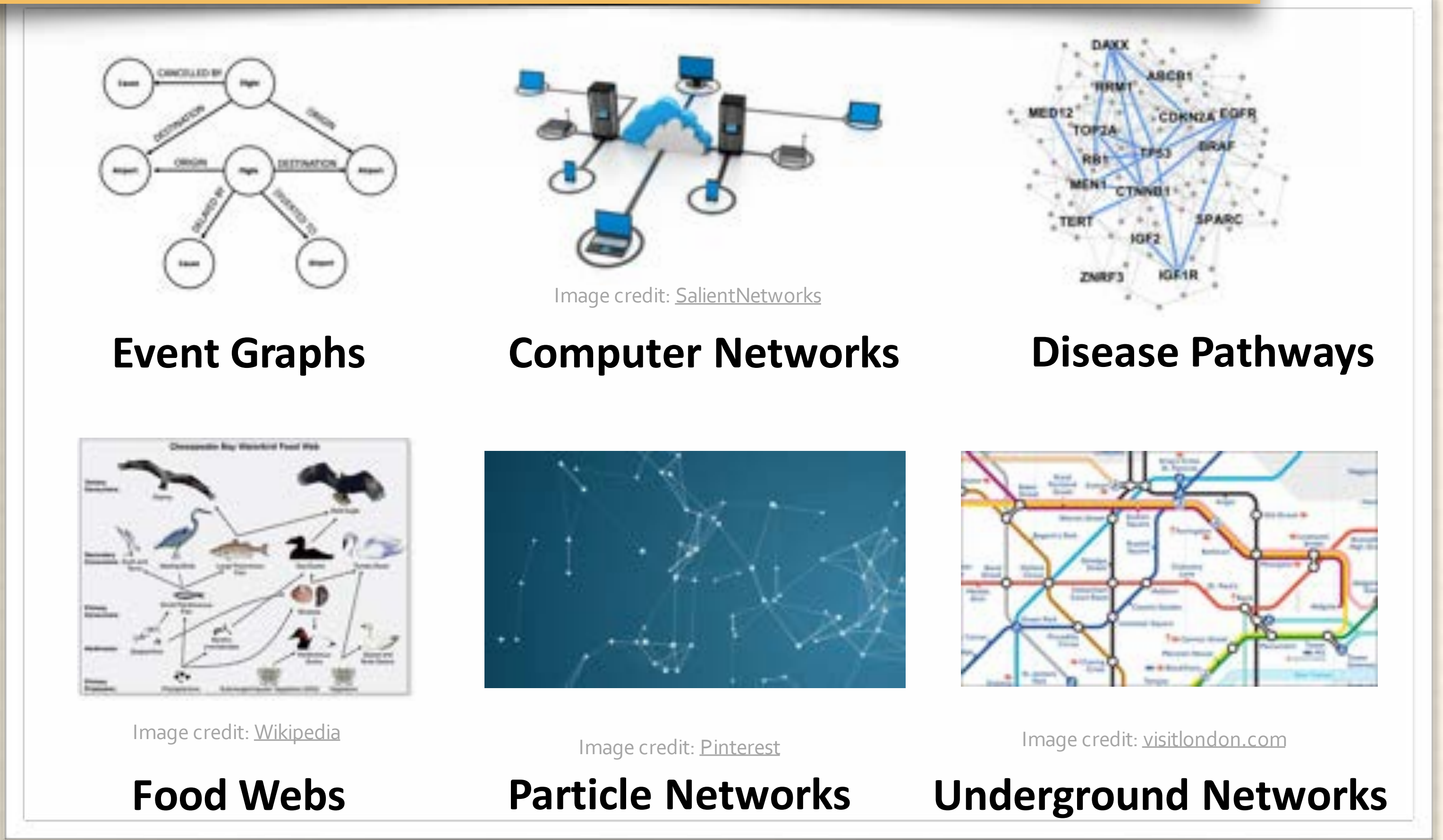


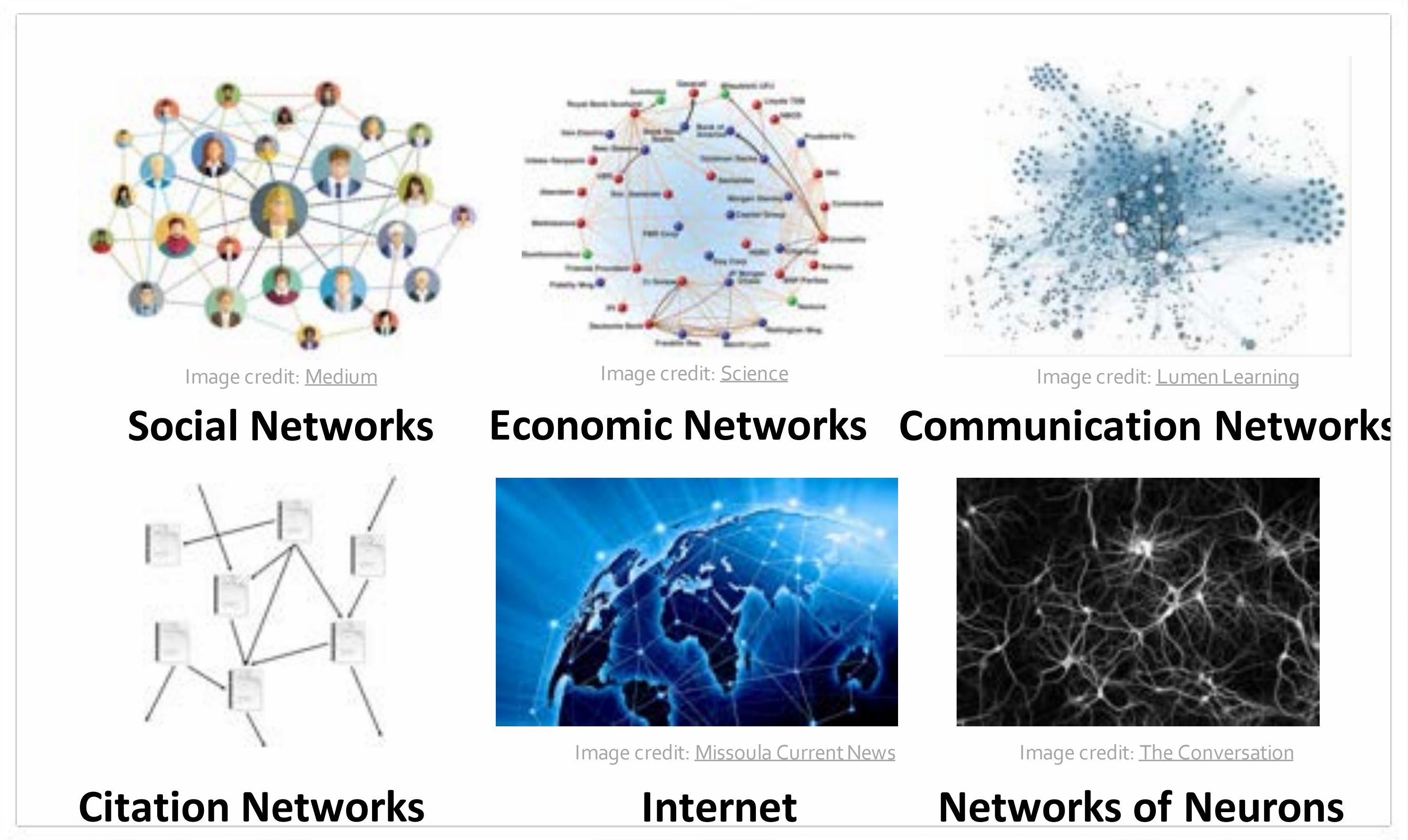
Graph learning

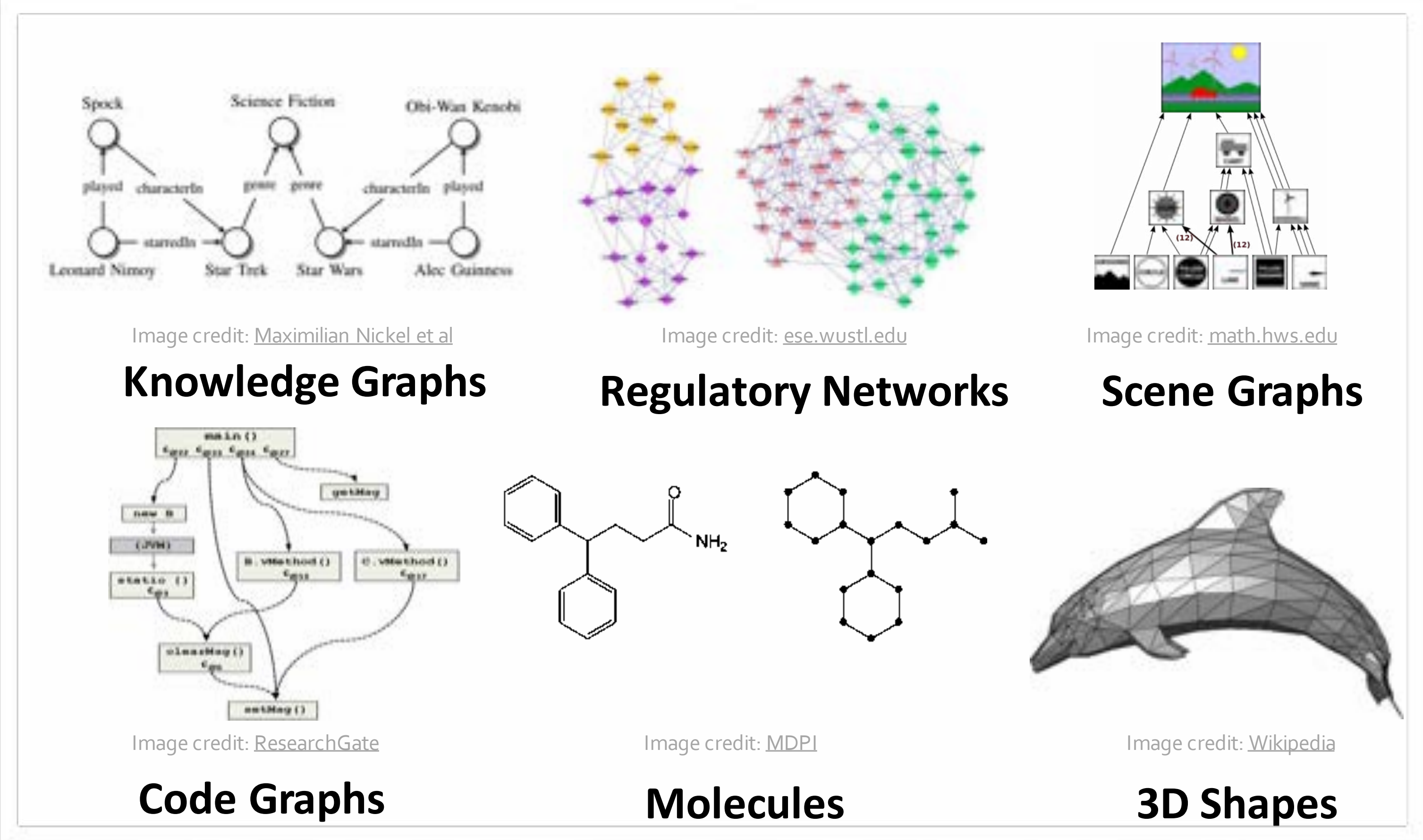
And other stuff

Why learning on graphs?

Graphs are everywhere!







Images: Machine Learning on Graphs,Course by Jure Leskovec

Graphs: One definition to rule them all

- Graph $G = (V_G, E_G, L_G)$ with
  - Vertex set V_G
  - Edge set $E_G \subseteq V_G^2 := V_G \times V_G$
  - Vertex labels: $L_G : V_G \to \Sigma$

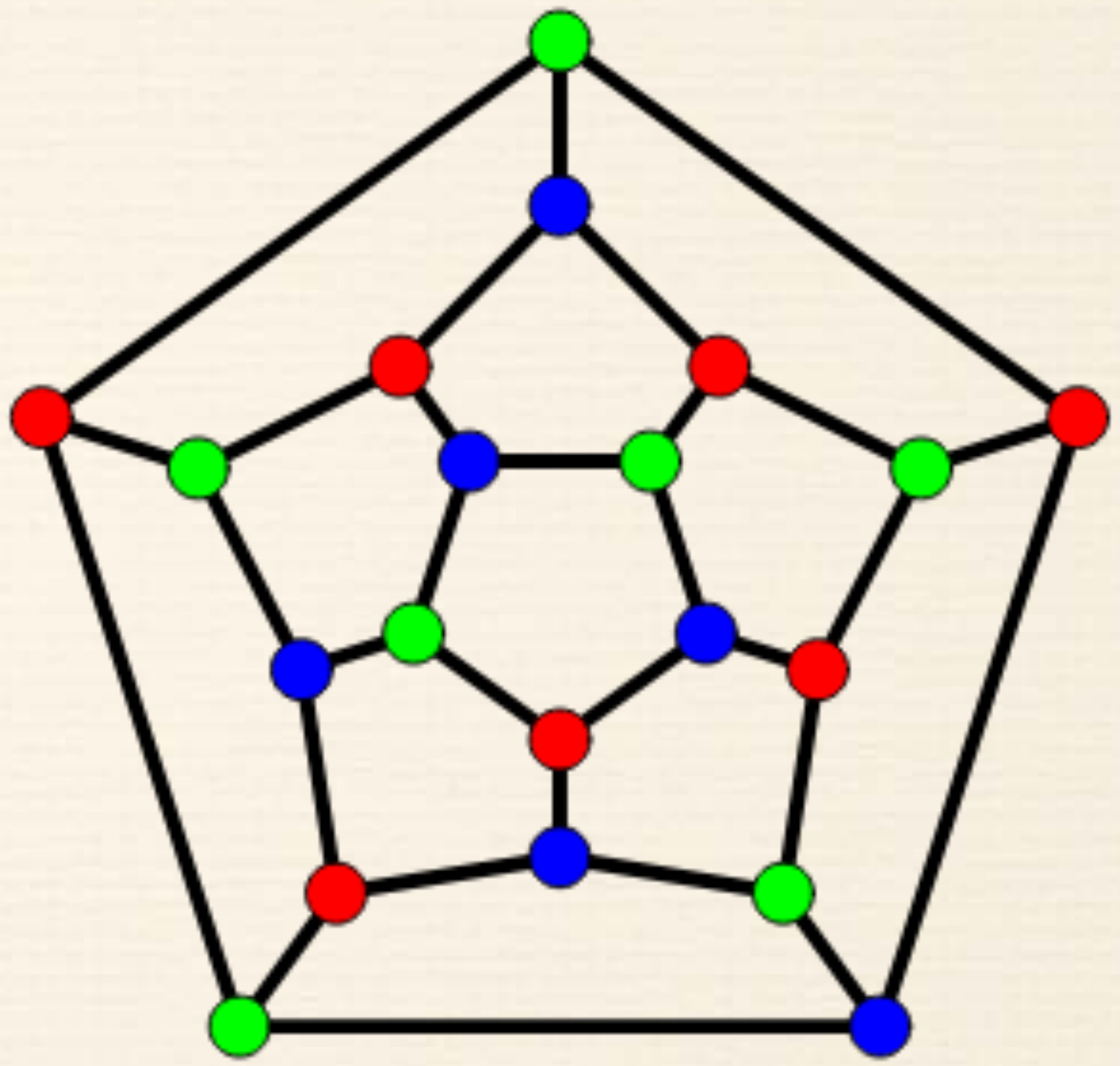
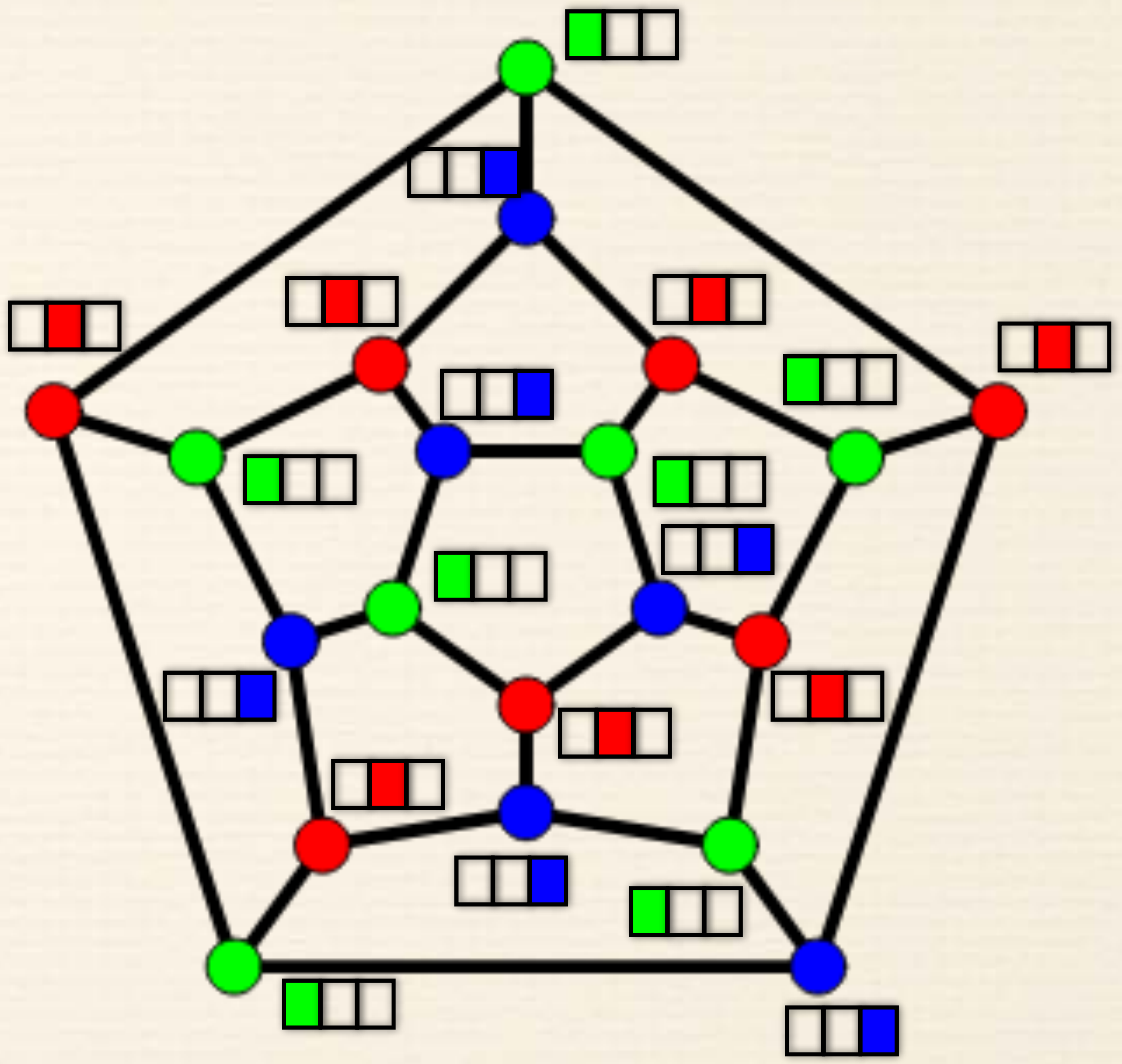


Image: Wikipedia

Graphs: One definition to rule them all

- Graph $G = (V_G, E_G, L_G)$ with
  - Vertex set V_G
  - Edge set $E_G \subseteq V_G^2 := V_G \times V_G$
  - Vertex labels: $L_G : V_G \to \Sigma$

Vertex features $\mathbb{R}^d$



Hot-one encoding

Image: Wikipedia

Adjacency matrix representation

- Graph $G = (V_G, E_G, L_G)$ can also be represented by adjacency matrix A_G and feature matrix F_G

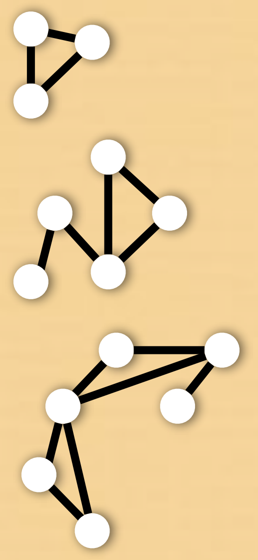
- Let $n = |V_G|$ be the number of vertices. Let $v, w \in [n] := \{1,\ldots,n\}$.

adjacency matrix $A_G \in \mathbb{R}^{n\times n} : (v,w) \mapsto \begin{cases} 1 & (v,w) \in E_G \\ 0 & \text{otherwise} \end{cases}$

feature matrix $F_G \in \mathbb{R}^{n\times d} : v \mapsto L_G(v)$

- Assumes an <u>ordering</u> on the vertices.

Graph learning



. . .

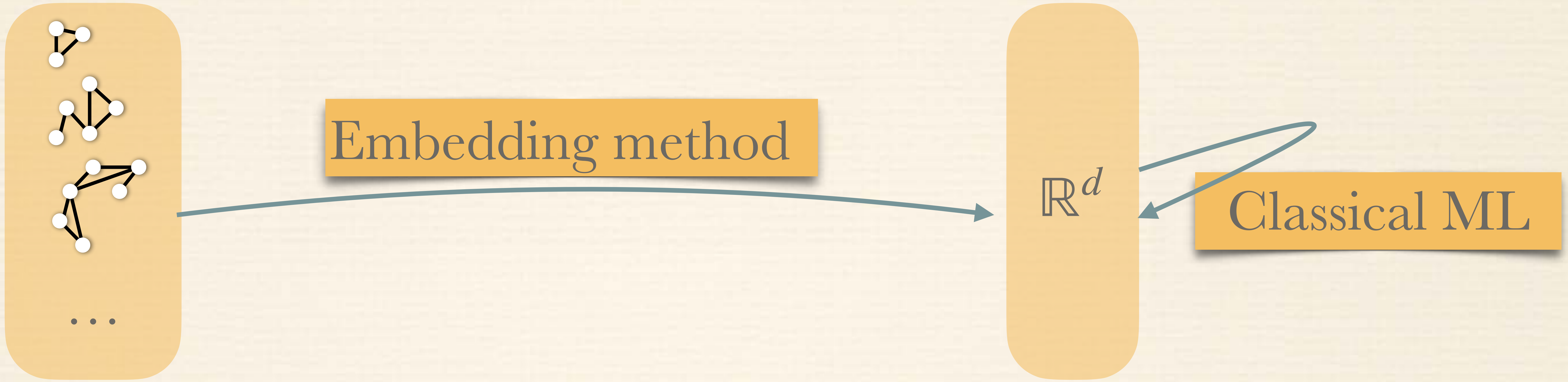
$\mathscr{G} =$ all graphs

$\mathbb{R}^d$

Classical ML

$\mathbb{Y} =$ output space

Graph learning



$\mathcal{G} =$ all graphs

$\mathbb{Y} =$ output space

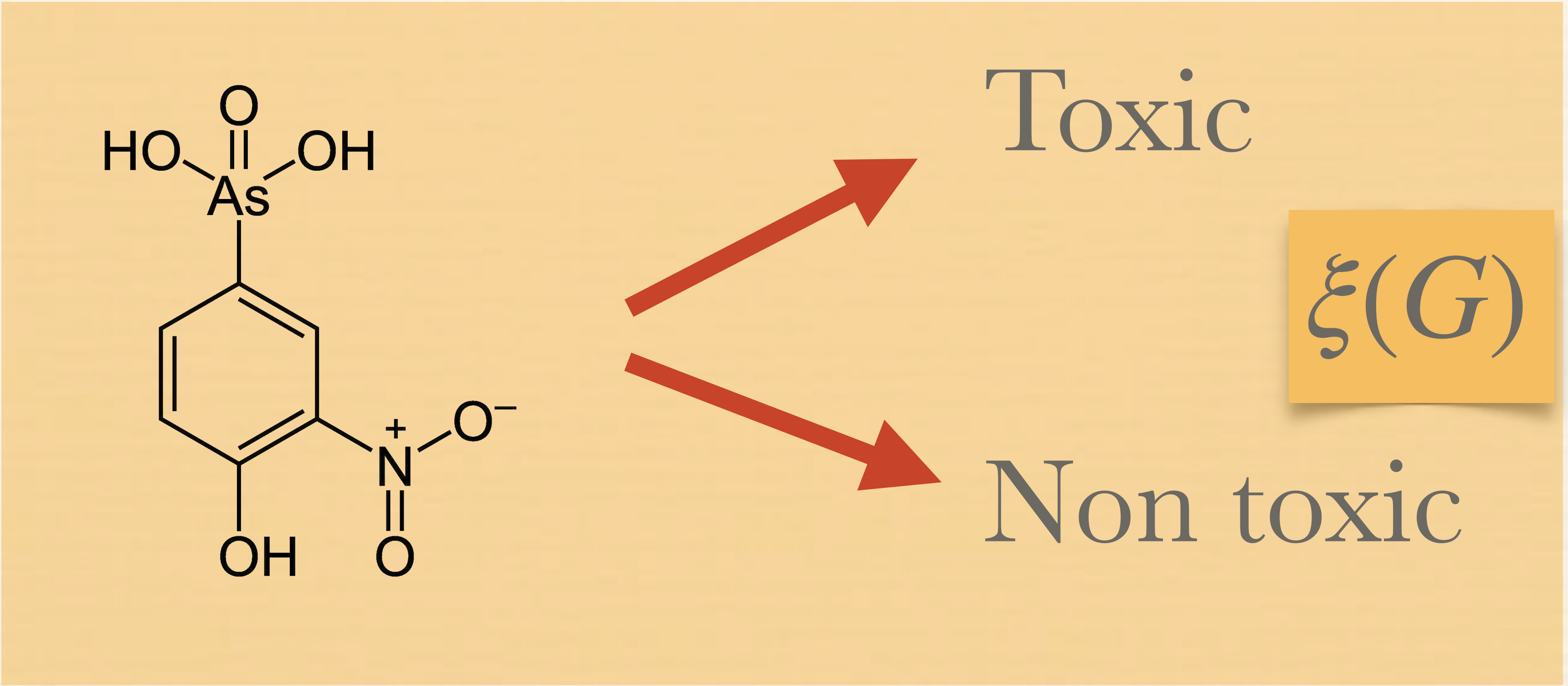
Embeddings

$$\begin{aligned} \mathscr{G} &= \text{ all graphs} \\ \mathscr{V} &= \text{ all vertices} \\ \mathbb{Y} &= \text{ output space} \end{aligned}$$

- Graph embedding: $\xi : \mathscr{G} \to \mathbb{Y}$
- Vertex embedding: $\xi : \mathscr{G} \to (\mathscr{V} \to \mathbb{Y})$
- p-Vertex embedding: $\xi : \mathscr{G} \to (\mathscr{V}^p \to \mathbb{Y})$

Graph embeddings

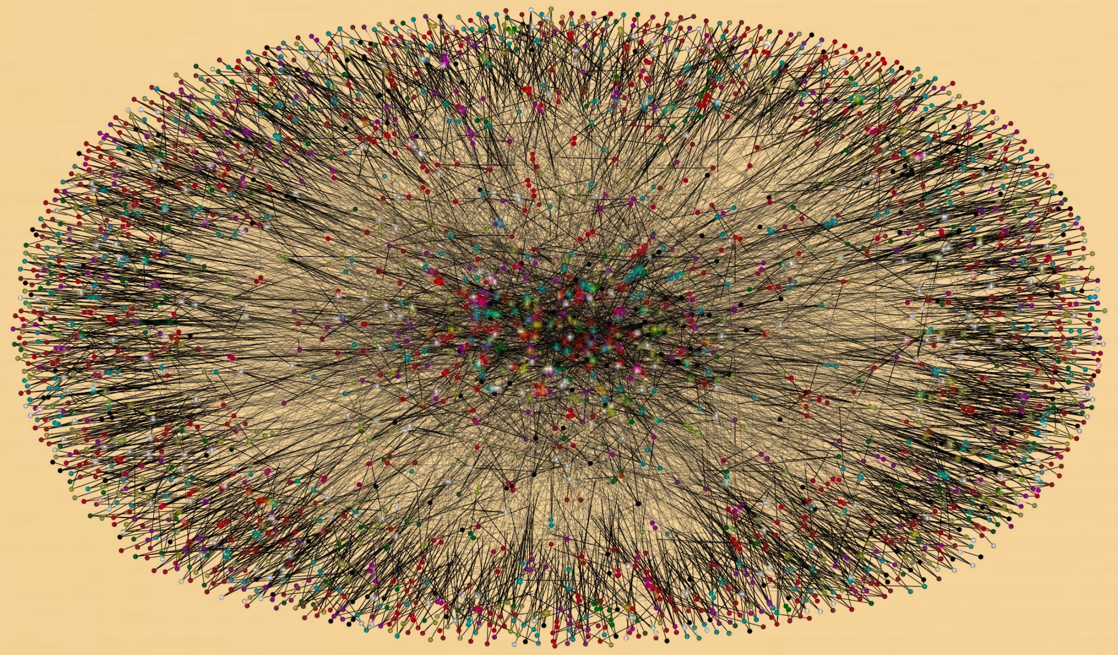
- Graph embedding: $\xi : \mathscr{G} \to \mathbb{Y}$
- Graph classification/regression



Vertex embeddings

- Vertex embedding: $\xi : \mathscr{G} \to (\mathscr{V} \to \mathbb{Y})$

- Vertex classification/regression. For example, prediction of subject of papers.



$$
\to \quad
\begin{array}{ccc}
 & & \xi(G, v) \\
\text{paper}_1 & \to & \text{math} \\
\text{paper}_2 & \to & \text{computer science} \\
\vdots & \vdots & \vdots \\
\text{paper}_n & \to & \text{biology}
\end{array}
$$

Images: Cora dataset

p-Vertex embeddings

- p-Vertex embedding: $\xi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y})$
- For example, 2-vertex embeddings: link prediction

$\xi(G, v, w)$

$$\to \begin{array}{lcl} (\text{Joe, Anna}) & \mapsto & \text{link} \\ (\text{Anastasios, Mohammed}) & \mapsto & \text{no link} \\ & \dots & \end{array}$$

Graph learning tasks

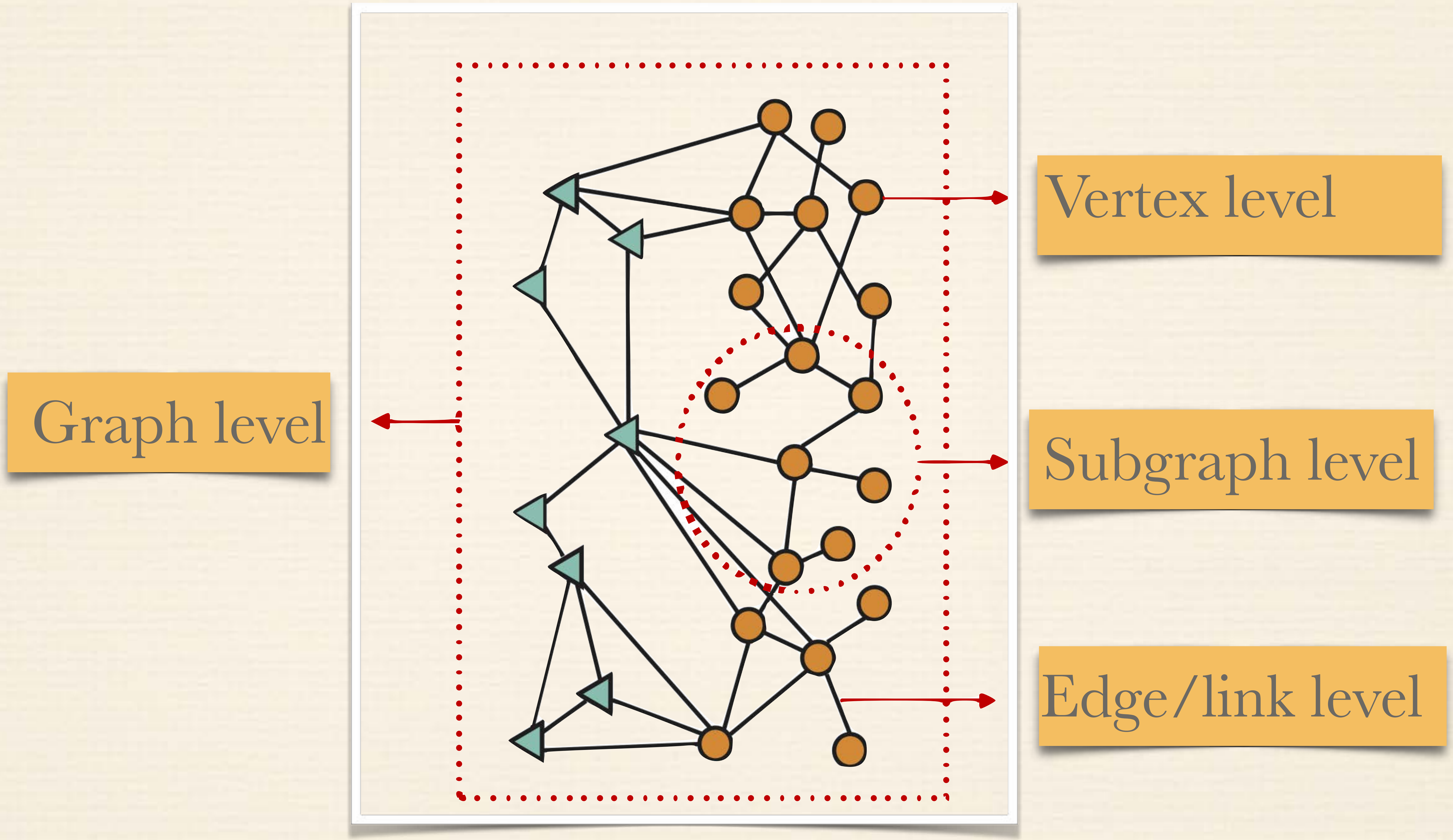
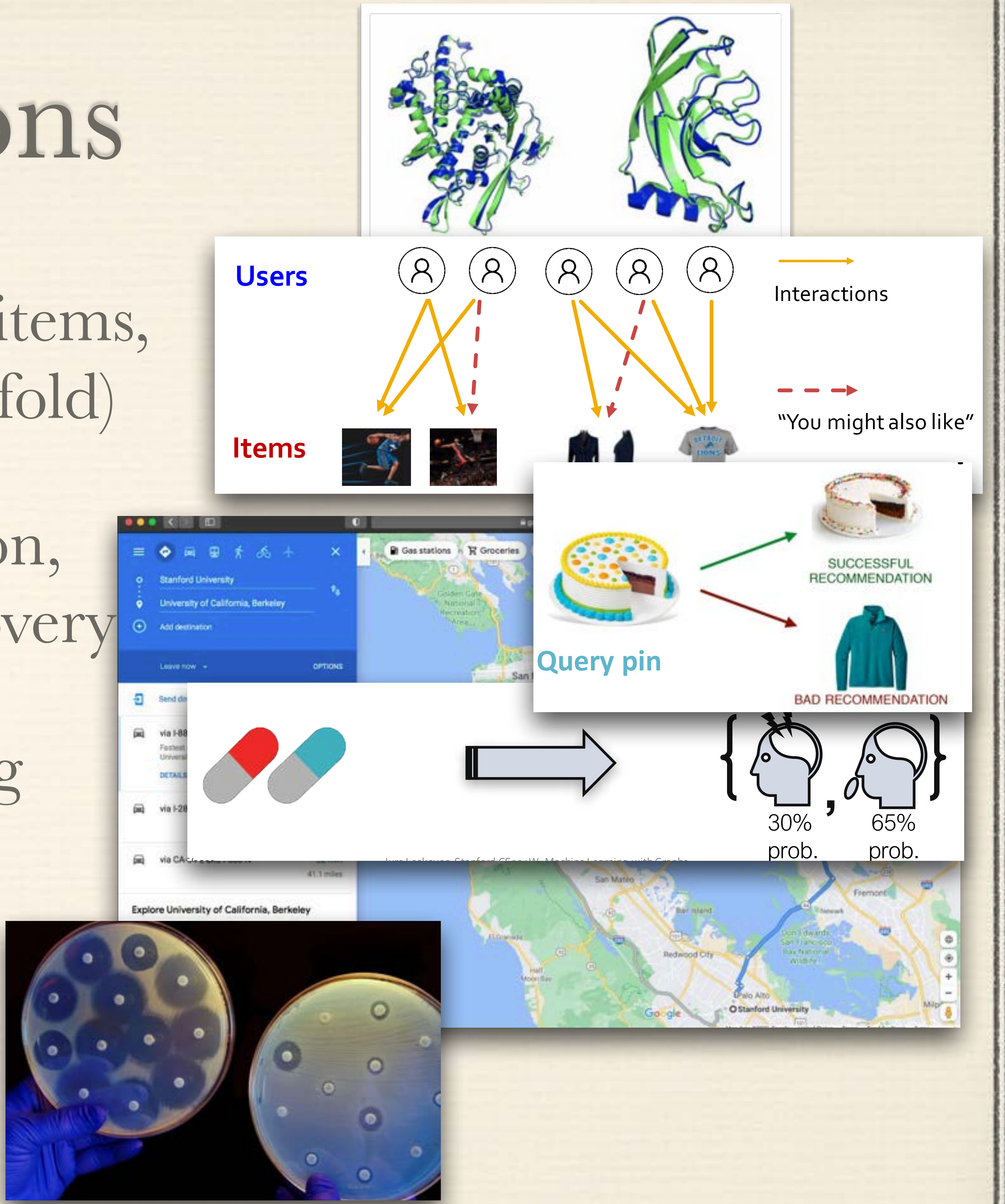


Image: Machine Learning on Graphs, course Jure Leskovec

Applications

- Vertex classification: categorise online user/items, location amino acids (protein folding, alpha fold)
- Link prediction: knowledge graph completion, recommender systems, drug side effect discovery
- Graph classification: molecule property, drug discovery
- Subgraph tasks: traffic prediction

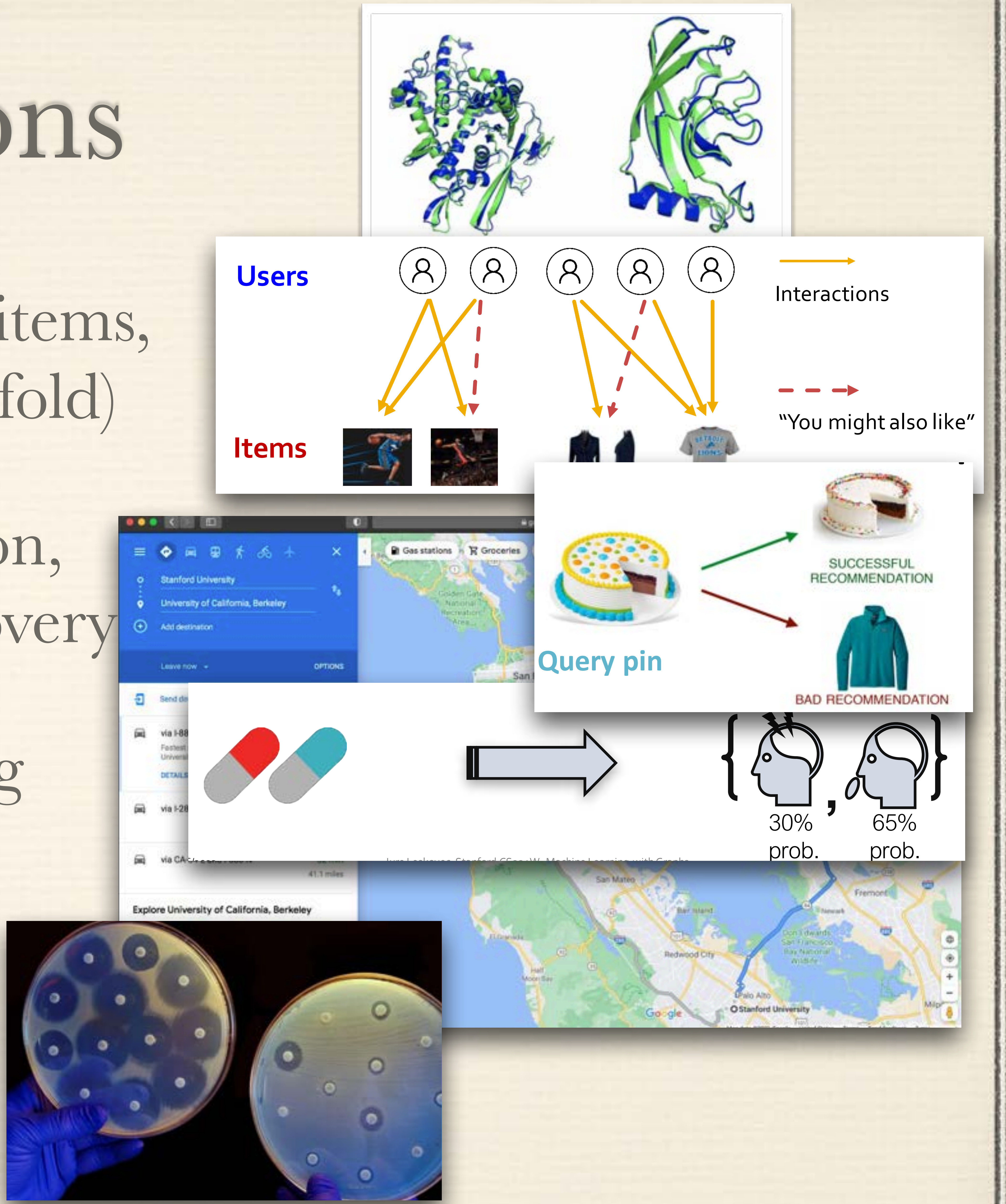


Images: Machine Learning on Graphs, course by Jure Leskovec

Applications

- Vertex classification: categorise online user/items, location amino acids (protein folding, alpha fold)
- Link prediction: knowledge graph completion, recommender systems, drug side effect discovery
- Graph classification: molecule property, drug discovery
- Subgraph tasks: traffic prediction

GRAPH LEARNING HAS BECOME KEY DATA SCIENCE COMPONENT



Images: Machine Learning on Graphs, course by Jure Leskovec

Graph learning

- We want to learn an unknown embedding $\Xi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y})$

Graph learning

- We want to learn an <u>unknown</u> embedding $\Xi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y})$

What does this mean???

Graph learning

- We want to learn an unknown embedding $\Xi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y})$

What does this mean???

- The embedding Ξ is partially revealed by means of a training set

$$\mathcal{T} := \{(G_1, \mathbf{v}_1, y_1), \ldots, (G_\ell, \mathbf{v}_\ell, y_\ell)\} \subseteq \mathcal{G} \times \mathcal{V}^p \times \mathbb{Y}$$

Graph learning

- We want to learn an unknown embedding $\Xi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y})$

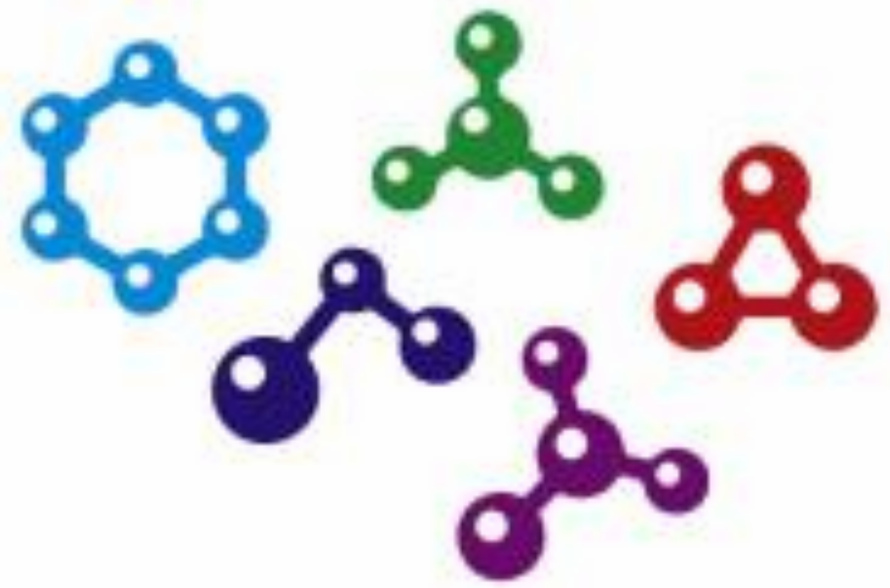
What does this mean???

- The embedding Ξ is partially revealed by means of a training set

$$\mathcal{T} := \{(G_1, \mathbf{v}_1, y_1), \ldots, (G_\ell, \mathbf{v}_\ell, y_\ell)\} \subseteq \mathcal{G} \times \mathcal{V}^p \times \mathbb{Y}$$

$\Xi(G_1, \mathbf{v}_1)$ $\Xi(G_\ell, \mathbf{v}_\ell)$

Training sets

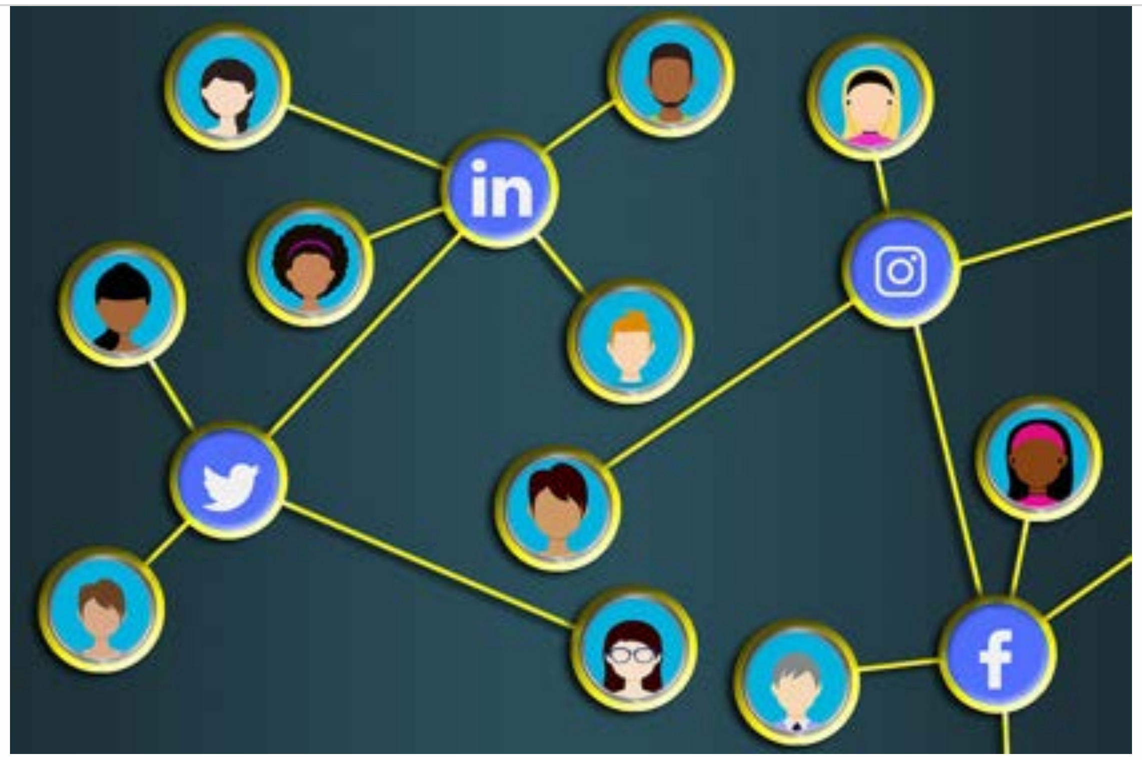


(molecule,yes/no)

Graph classification

(cora, paper, topic)

Vertex classification



(social, p_x, p_y, yes/no)

Link prediction

Graph learning: hypothesis class

- We want to find the best model consistent with training set $\mathscr{T}$

Graph learning: hypothesis class

- We want to find the best model consistent with training set $\mathcal{T}$

What does this mean???

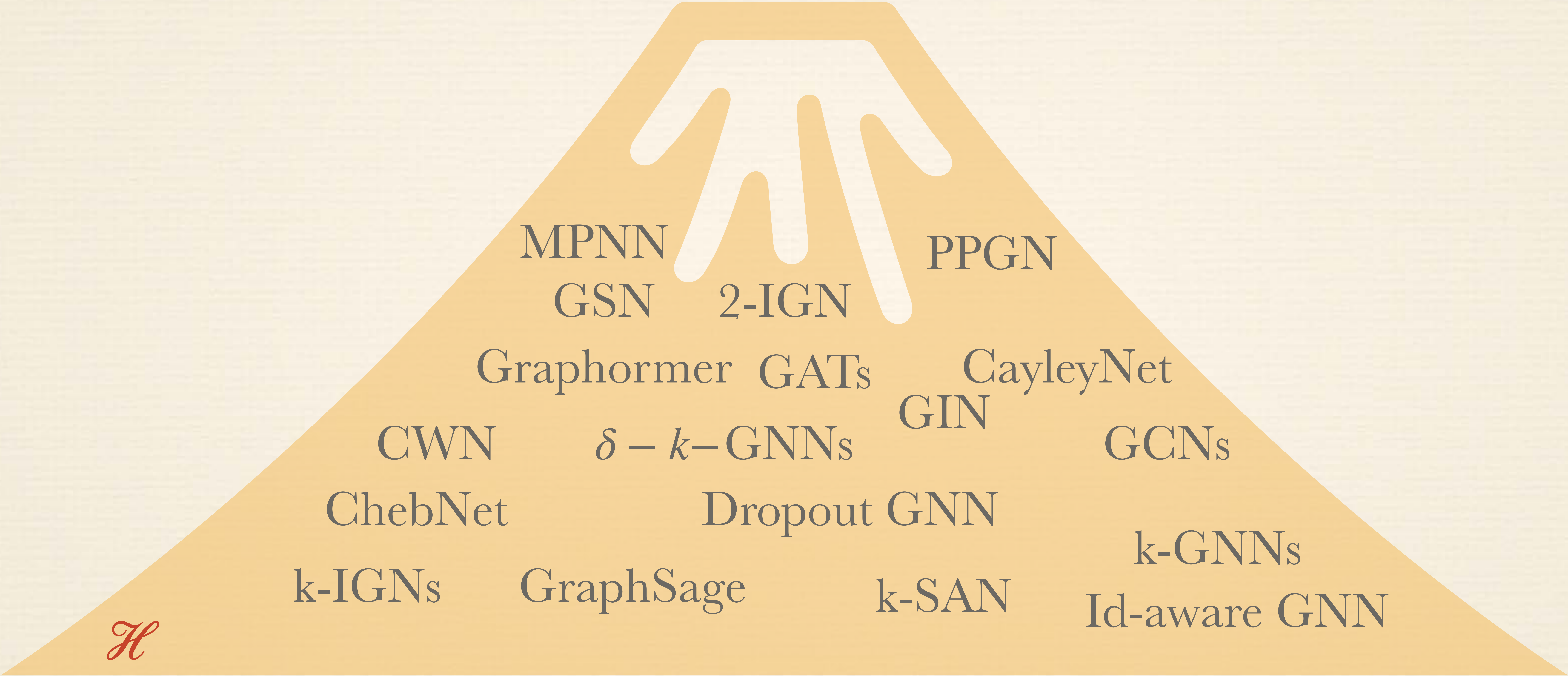
Graph learning: hypothesis class

- We want to find the best model consistent with training set $\mathscr{T}$

What does this mean???

- Models are selected from an hypothesis class $\mathscr{H}$
- In the graph setting $\mathscr{H}$ consists of embeddings

Hypothesis classes



Hypothesis classes

MPNN

PPGN

GSN 2-IGN

×1000

Graphormer GATs CayleyNet

GIN

CWN $\delta - k-$GNNs GCNs

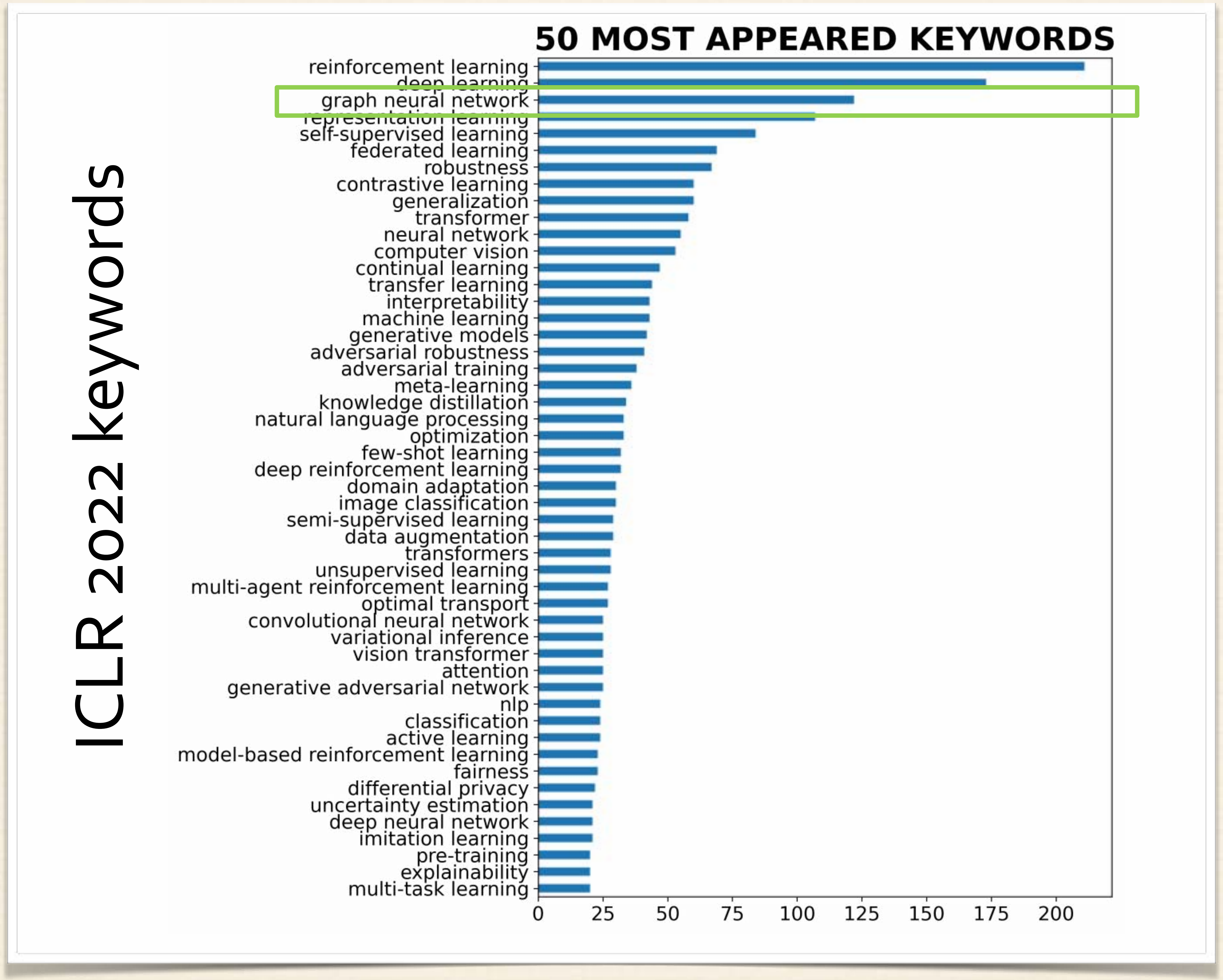
ChebNet Dropout GNN

k-GNNs

k-IGNs GraphSage k-SAN Id-aware GNN

$\mathcal{H}$

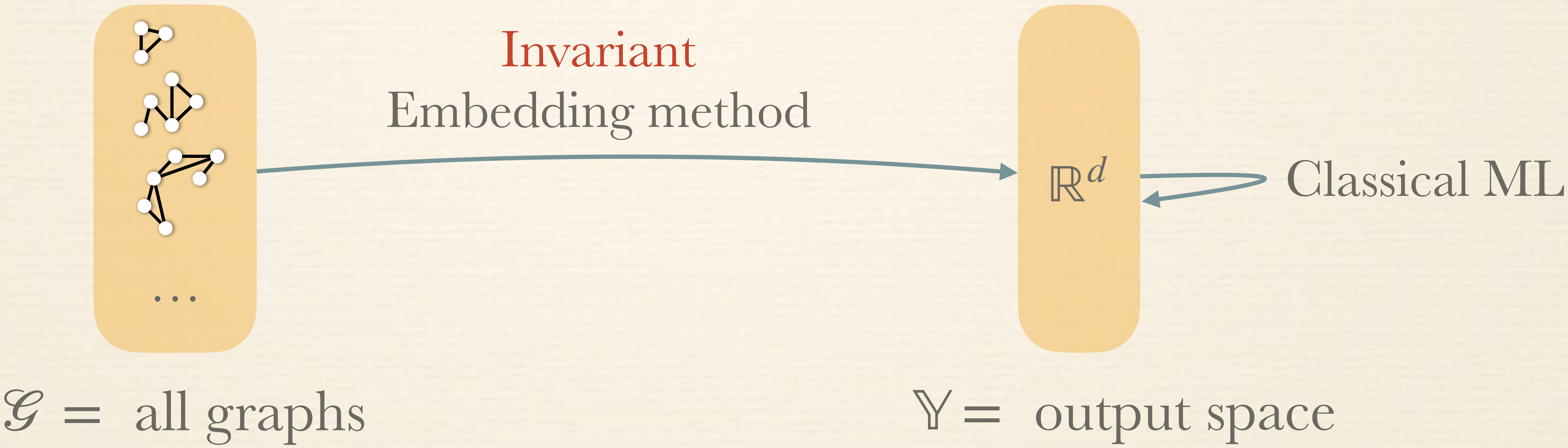
Explosion



Images: Machine Learning on Graphs, course J. Leskovec

Graph learning

Classical embedding methods depend on representation
E.g., think of MLP on vector representation of flattened adjacency matrix



A desired property: Invariance

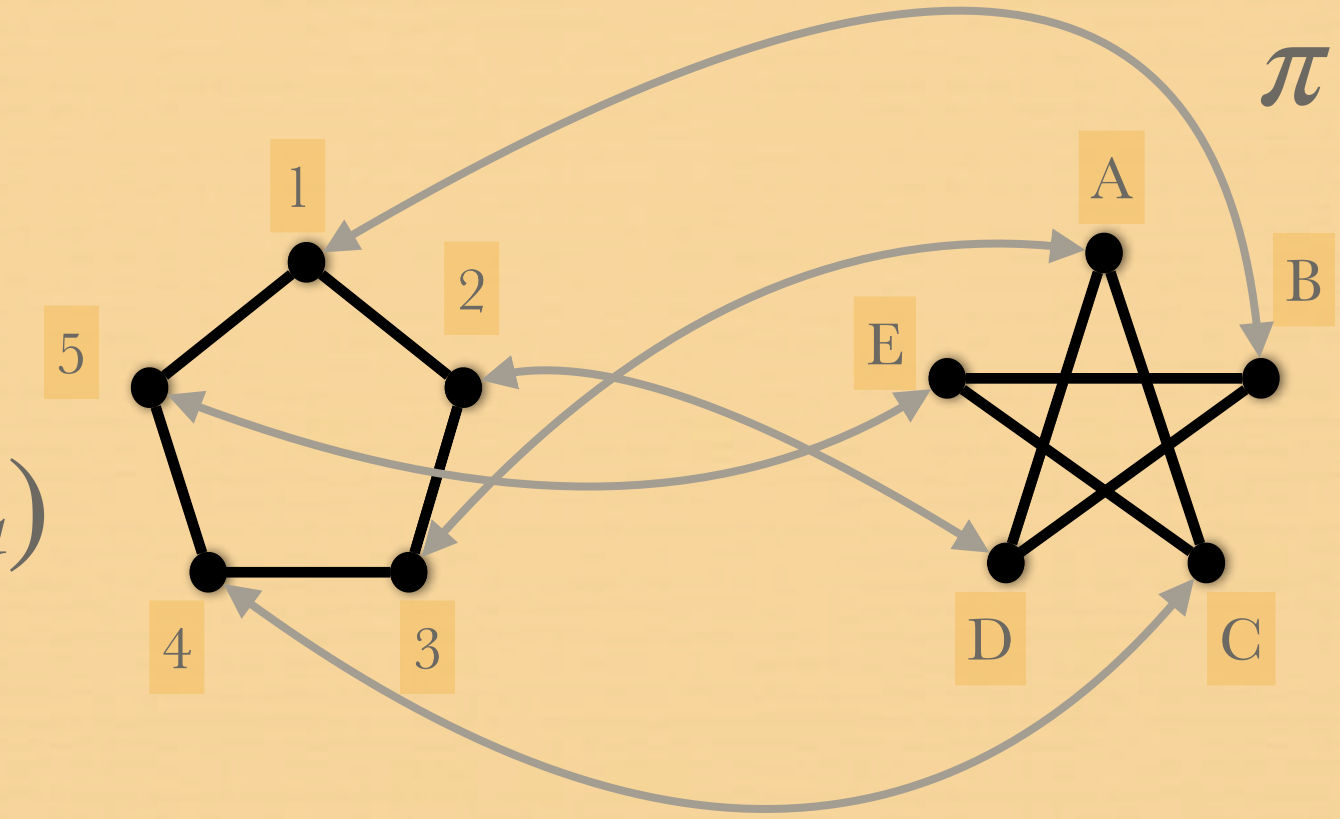
- Embeddings should be invariant, that is, independent of the chosen graph representation.

- Invariance is defined in terms of graph isomorphisms.

$G \cong H$

$\pi : V_G \to V_H$

$G = (V_G, E_G)$



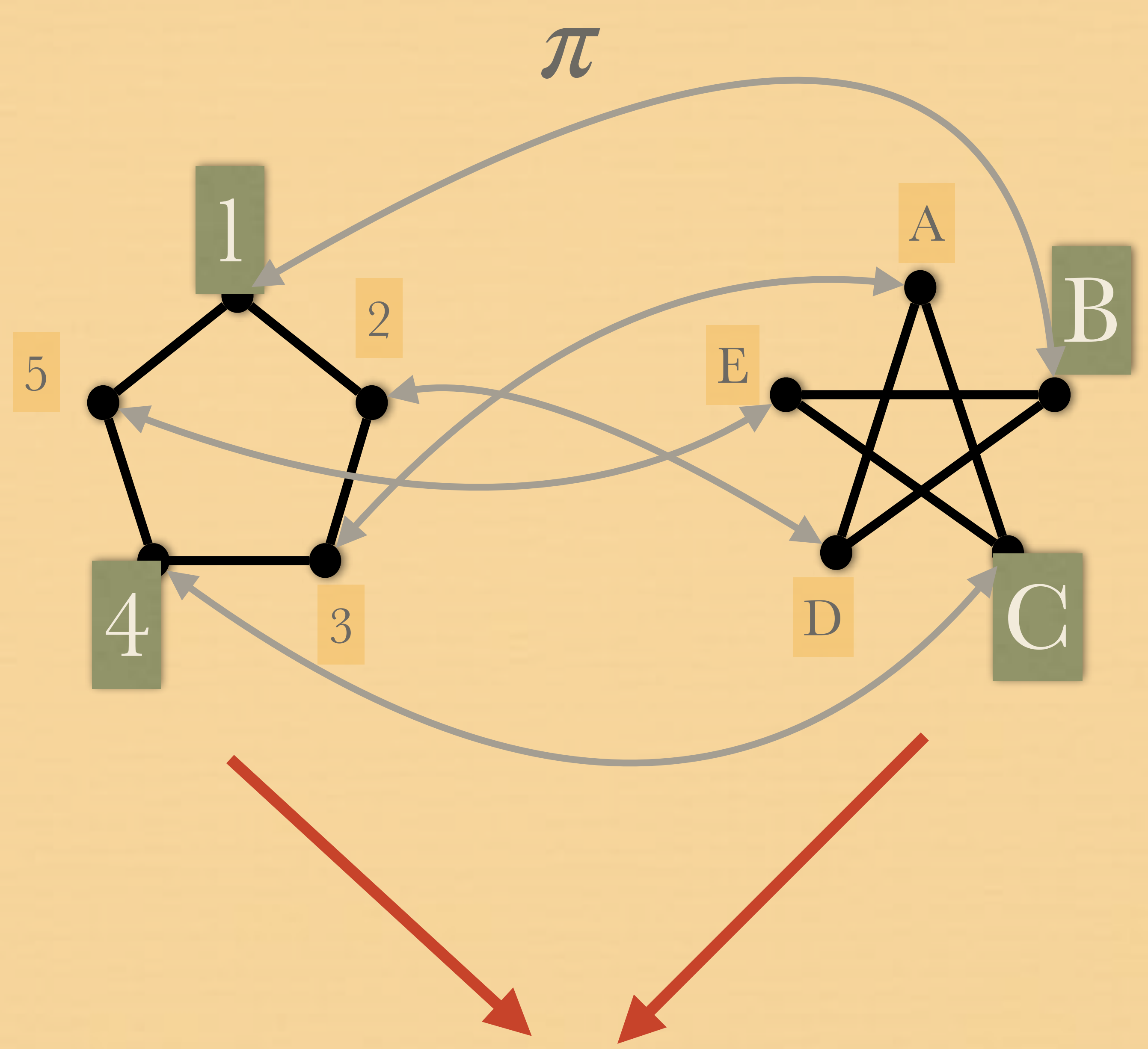
$H = (V_H, E_H)$

- The mapping π is a bijective vertex function satisfying $(v, v') \in E_G \iff (\pi(v), \pi(w)) \in E_H$ also $L_G(v) = L_H(\pi(v))$ must hold.

Invariant embeddings

for all π, G and $\mathbf{v} \in V_G^p : \xi(G, \mathbf{v}) = \xi(\pi(G), \pi(\mathbf{v}))$

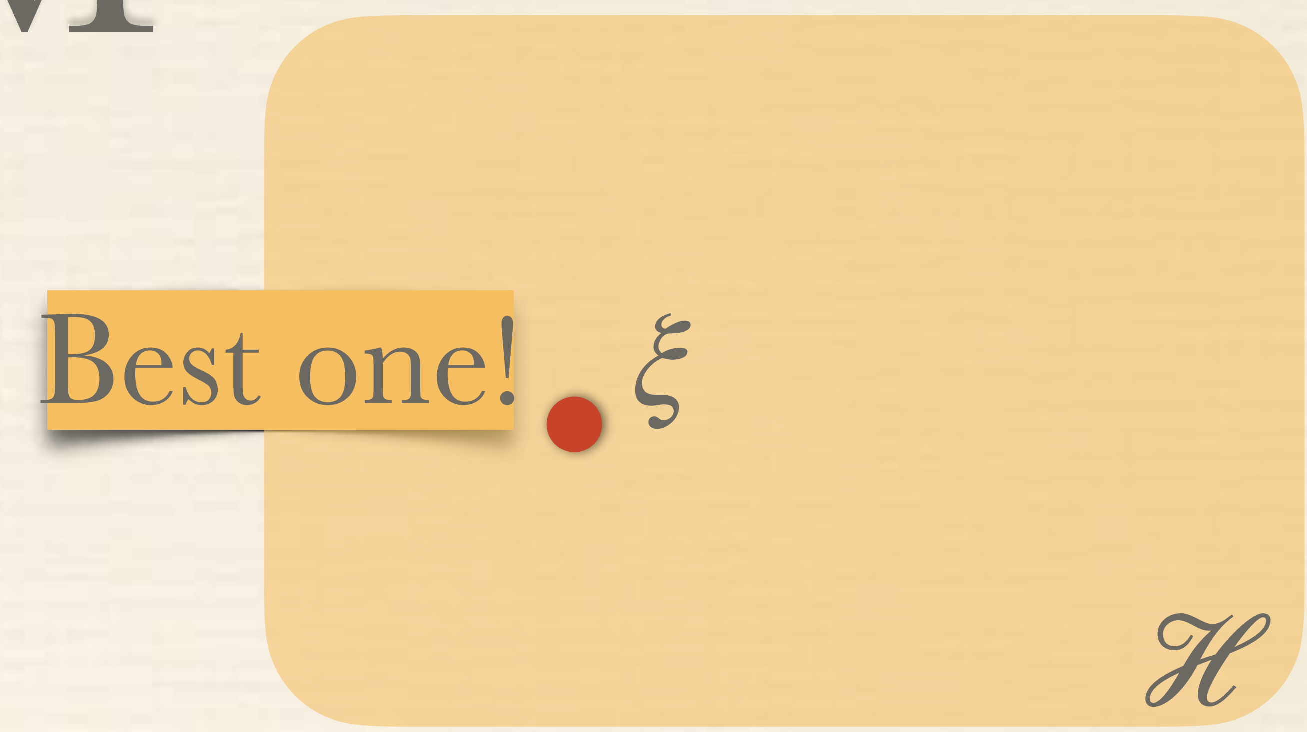
Isomorphism



(1,4) and (B,C) have same embedding in $\mathbb{Y}$

We typically assume invariant embedding methods (unless said otherwise)

Graph learning: ERM



- Given training set $\mathscr{T}$ and hypothesis class $\mathscr{H}$
- Empirical risk minimisation:

Find embedding ξ in $\mathscr{H}$ which minimises empirical loss

$$\frac{1}{\ell}\sum_{i=1}^{\ell} \mathsf{loss}(\xi(G_i, \mathbf{v}_i), y_i))$$

Loss function is a mapping from $\mathbb{Y} \times \mathbb{Y} \rightarrow \mathbb{R}$

Loss functions

- Choice depends on learning task (regression, classification,…)
  - L1: $\text{loss}(y_{predicted}, y_{true}) := |y_{predicted} - y_{true}|$
  - L2: $\text{loss}(y_{predicted}, y_{true}) := (y_{predicted} - y_{true})^2$
  - (Binary) cross entropy:
    $\text{loss}(y_{predicted}, y_{true}) := y_{true}\log(y_{predicted} + (1 - y_{true})\log(1 - y_{predicted})$

Graph learning

- Graph learning systems solve ERM using back propagation and gradient descent…

$$\hat{\xi} : \arg\min_{\xi \in \mathcal{H}} \frac{1}{\ell} \sum_{i=1}^{\ell} \mathsf{loss}(\xi(G_i, \mathbf{v}_i), y_i))$$

Graph learning

- Graph learning systems solve ERM using back propagation and gradient descent…

$$\hat{\xi} : \arg\min_{\xi \in \mathcal{H}} \frac{1}{\ell} \sum_{i=1}^{\ell} \mathsf{loss}(\xi(G_i, \mathbf{v}_i), y_i))$$

Our focus will be on the expressive power of hypothesis classes

Expressive power

- Which embeddings can be expressed by embeddings in $\mathcal{H}$?
- Which embeddings can be approximated by embeddings in $\mathcal{H}$?
- Which inputs can be separated/distinguished by embeddings in $\mathcal{H}$?

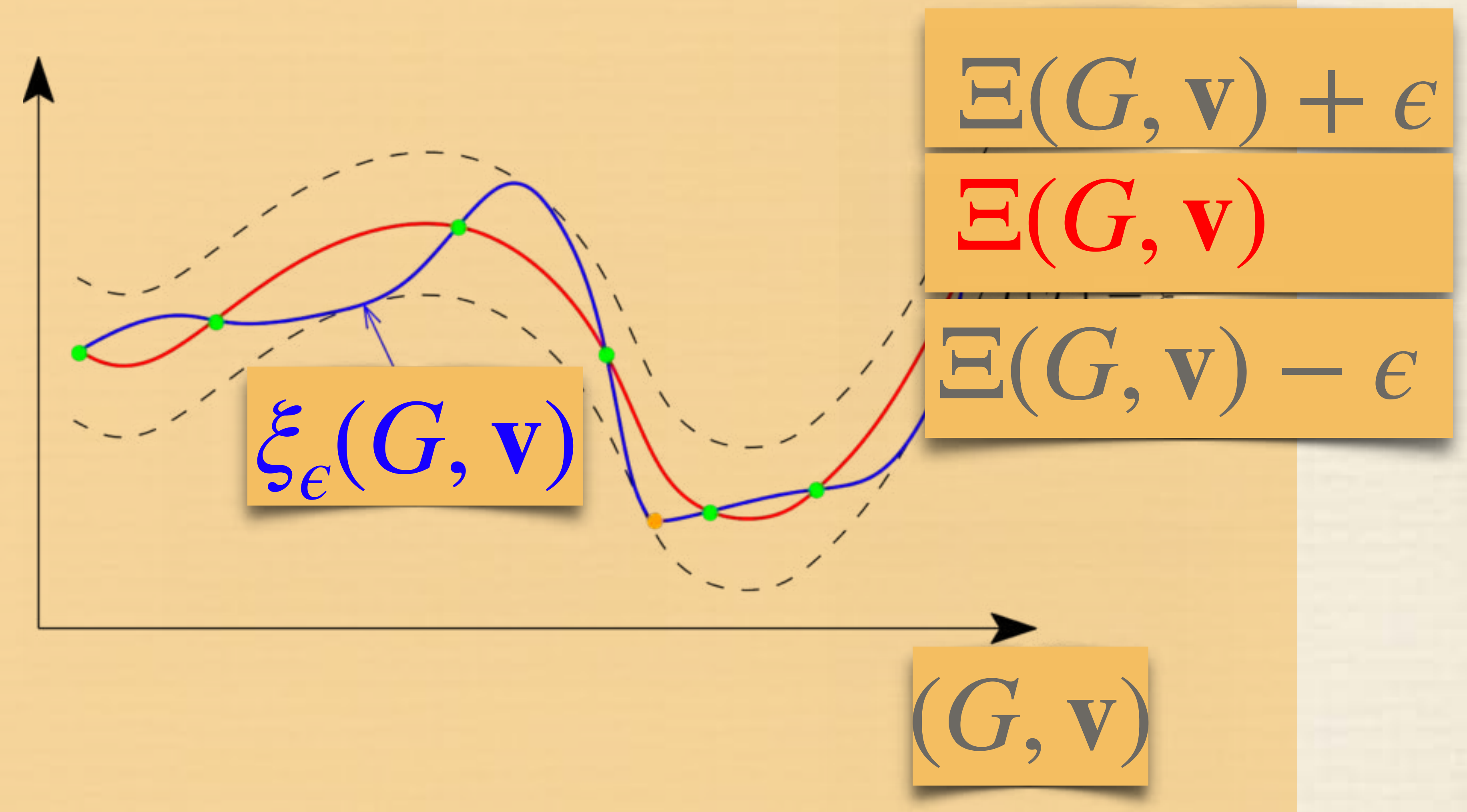
Notions of expressivity I

- Let $\Xi : \mathscr{G} \to (\mathscr{V}^p \to \mathbb{Y})$ be a p-vertex embedding and let $\mathscr{C}$ be a subset of $\mathscr{G}$

$\mathscr{H}$ can $\mathscr{C}$-express Ξ if there exists a $\xi \in \mathscr{H}$ such that for all $G \in \mathscr{C}, \mathbf{v} \in V_G^p$:

$\xi(G, \mathbf{v}) = \Xi(G, \mathbf{v})$

$\mathscr{H}$ can $\mathscr{C}$-approximate Ξ if for any $\epsilon > 0$ there exists a $\xi_\epsilon \in \mathscr{H}$ such that for all $G \in \mathscr{C}, \mathbf{v} \in V_G^p$: $\|\xi_\epsilon(G, \mathbf{v}) - \Xi(G, \mathbf{v})\| \leq \epsilon$



Notions of expressivity II

Separation/distinguishing power of $\mathcal{H}$

$$\rho(\mathcal{H}) := \{(G, \mathbf{v}, H, \mathbf{w}) \mid \forall \xi \in \mathcal{H} : \xi(G, \mathbf{v}) = \xi(H, \mathbf{w})\}$$

- All pairs of inputs that cannot be separated by any embedding in $\mathcal{H}$

Distinguishing power

- Strongest power: $\mathscr{H}$ powerful enough to detect non-isomorphic graphs

- Weakest power: $\mathscr{H}$ cannot differentiate any two graphs

Distinguishing power

- Allows for comparing different classes of embeddings methods!

$$\rho(\text{methods1}) \subseteq \rho(\text{methods2})$$

Methods1 is more powerful than Methods2
Methods 2 is bounded by Methods 1 in power

$$\rho(\text{methods1}) = \rho(\text{methods2})$$

Both methods are as powerful

- Allows for comparing embedding methods with algorithms, logic, …

Distinguishing power

- Allows for comparing different classes of embeddings methods!

$$\rho(\text{methods1}) \subseteq \rho(\text{methods2})$$

Methods1 is more powerful than Methods2
Methods 2 is bounded by Methods 1 in power

$$\rho(\text{methods1}) = \rho(\text{methods2})$$

Both methods are as powerful

- Allows for comparing embedding methods with algorithms, logic, …

Expressive power in ML community

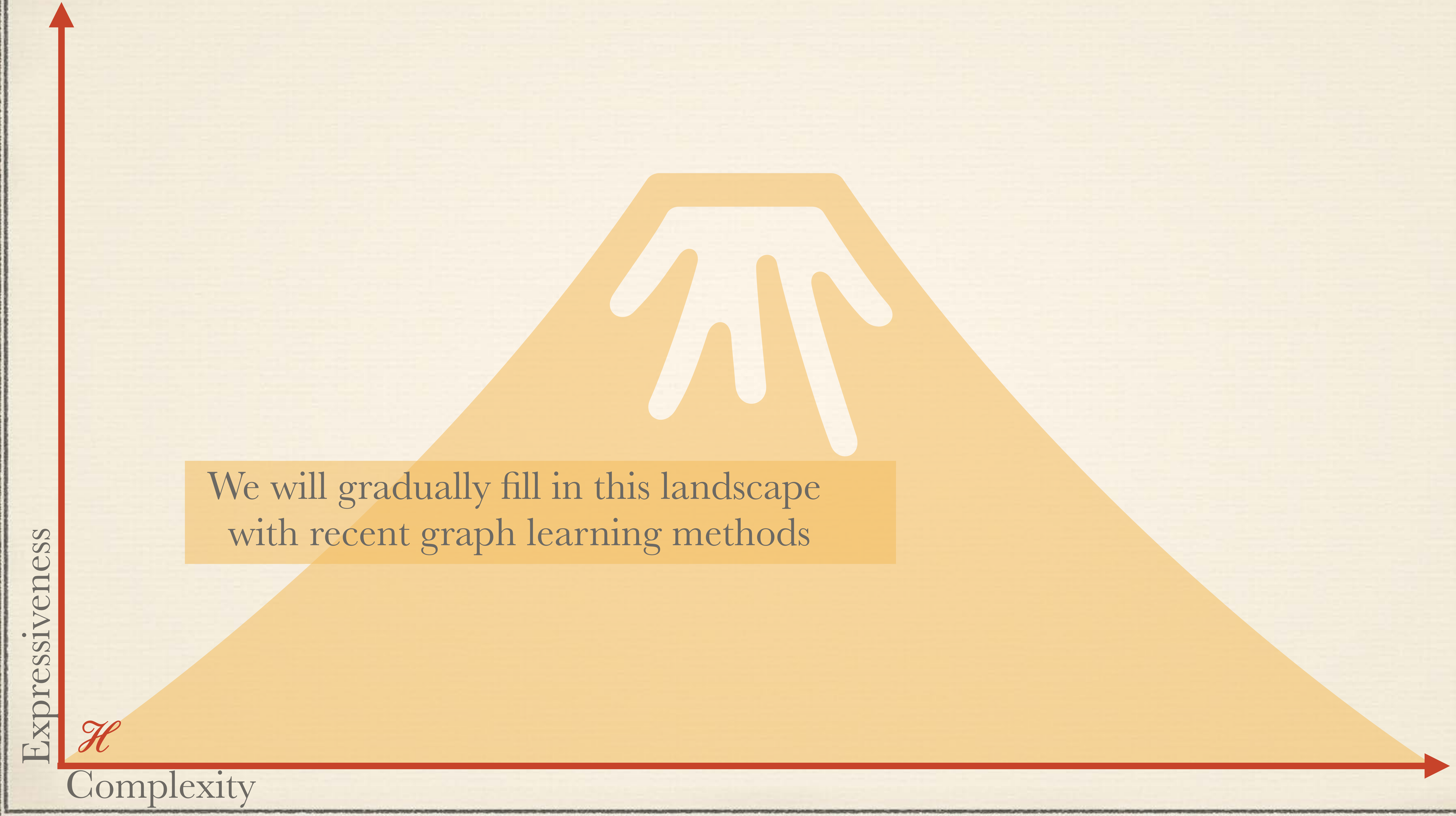
- Focus has been on distinguishing power of classes $\mathscr{H}$ of embedding methods.

- Goal is to characterise $\rho(\mathscr{H})$ in a way to sheds light on what graph properties a learning method can detect/use.

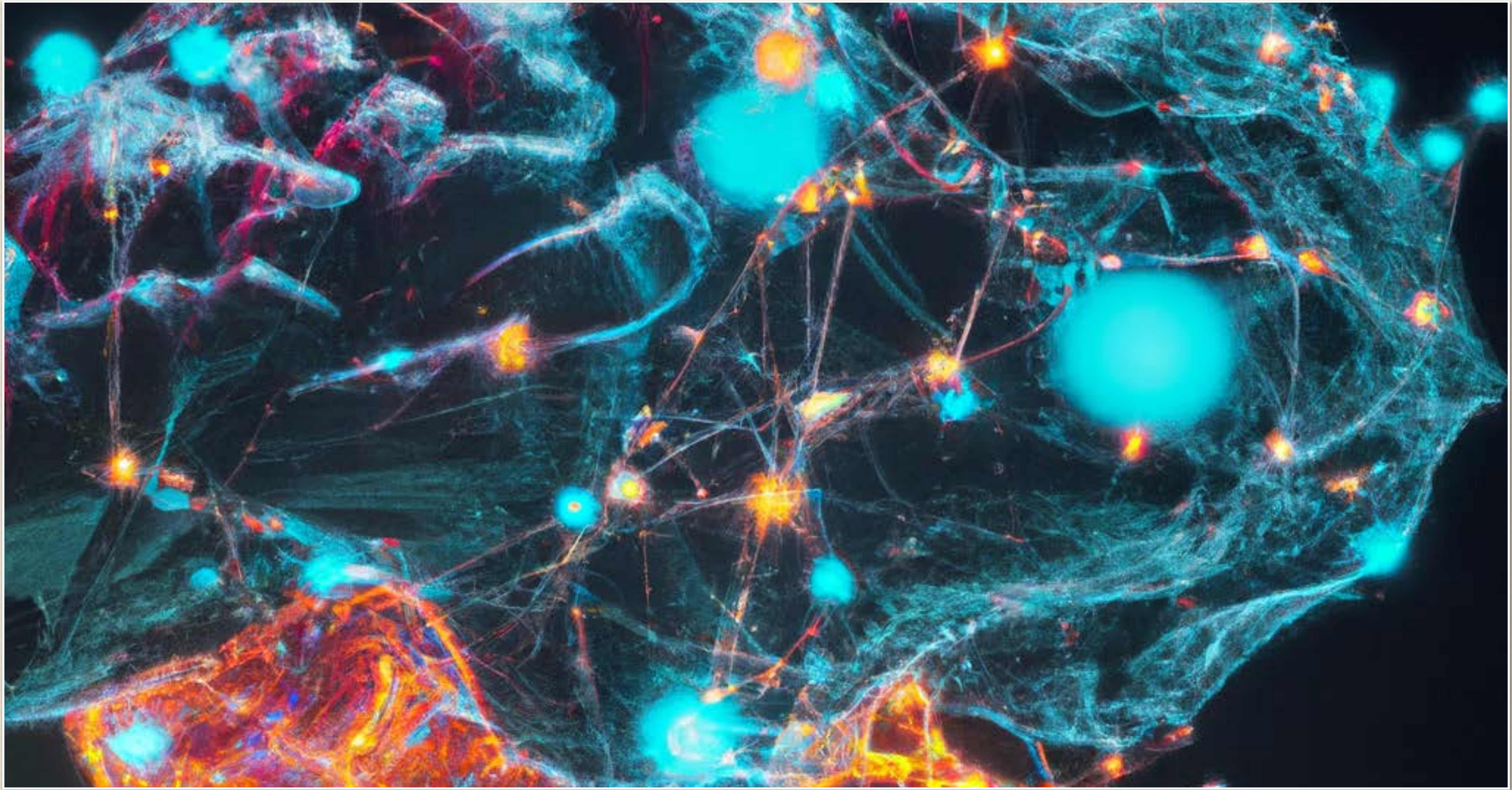
- We see an example shortly for $\mathscr{H}$ = the class of Message-Passing Neural Networks (MPNNs)

Expressive power in ML community

- Search for increase in expressive power has led to surge of new methods of graph learning.
- Despite theoretical underpinning… still a bit of alchemy to find the right method…


We will gradually fill in this landscape
with recent graph learning methods
Expressiveness
$\mathcal{H}$
Complexity

Questions?



Message Passing Neural Networks

The most popular type of GNNs

A little history

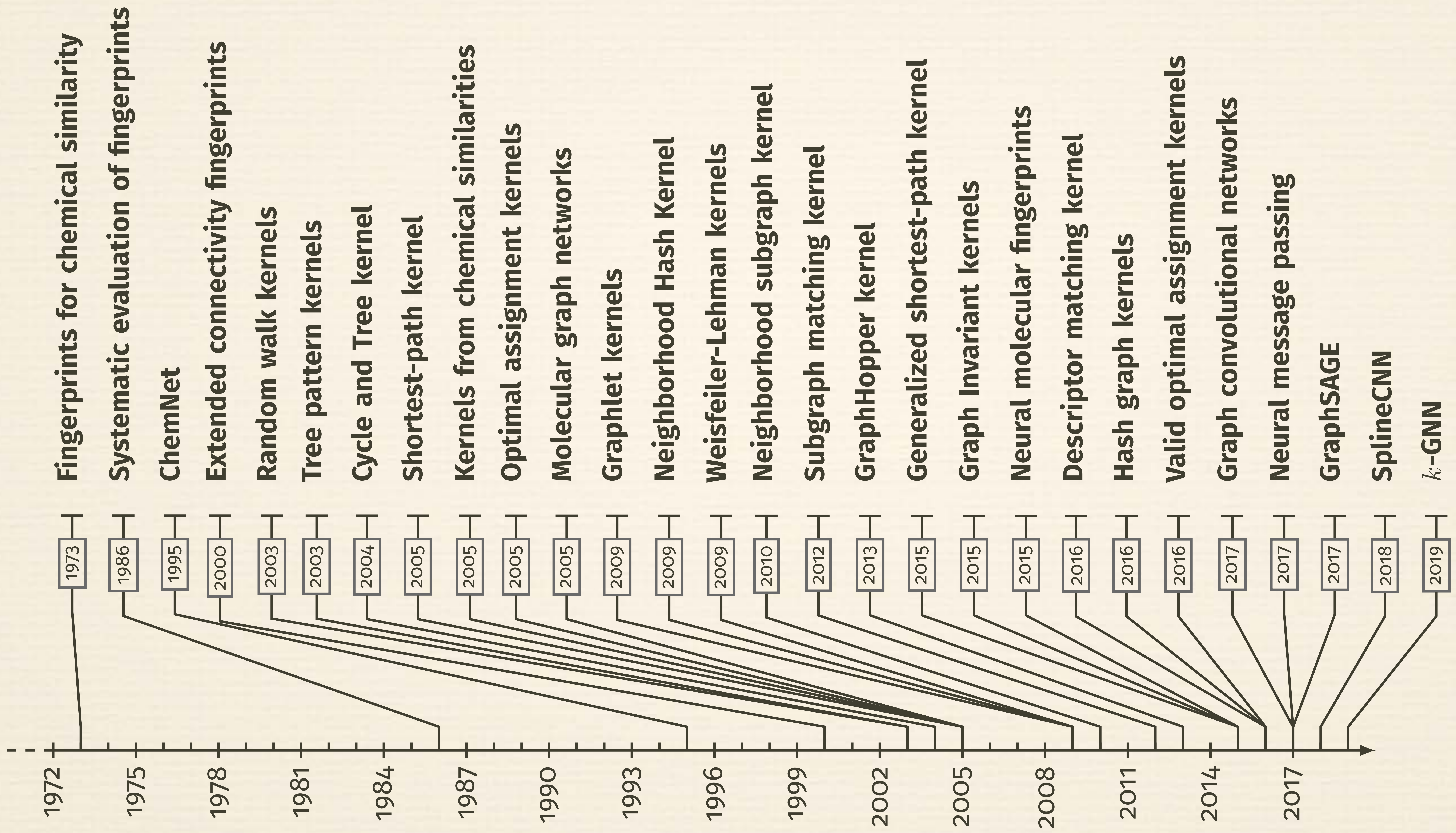


Image: Christopher Morris

A little history

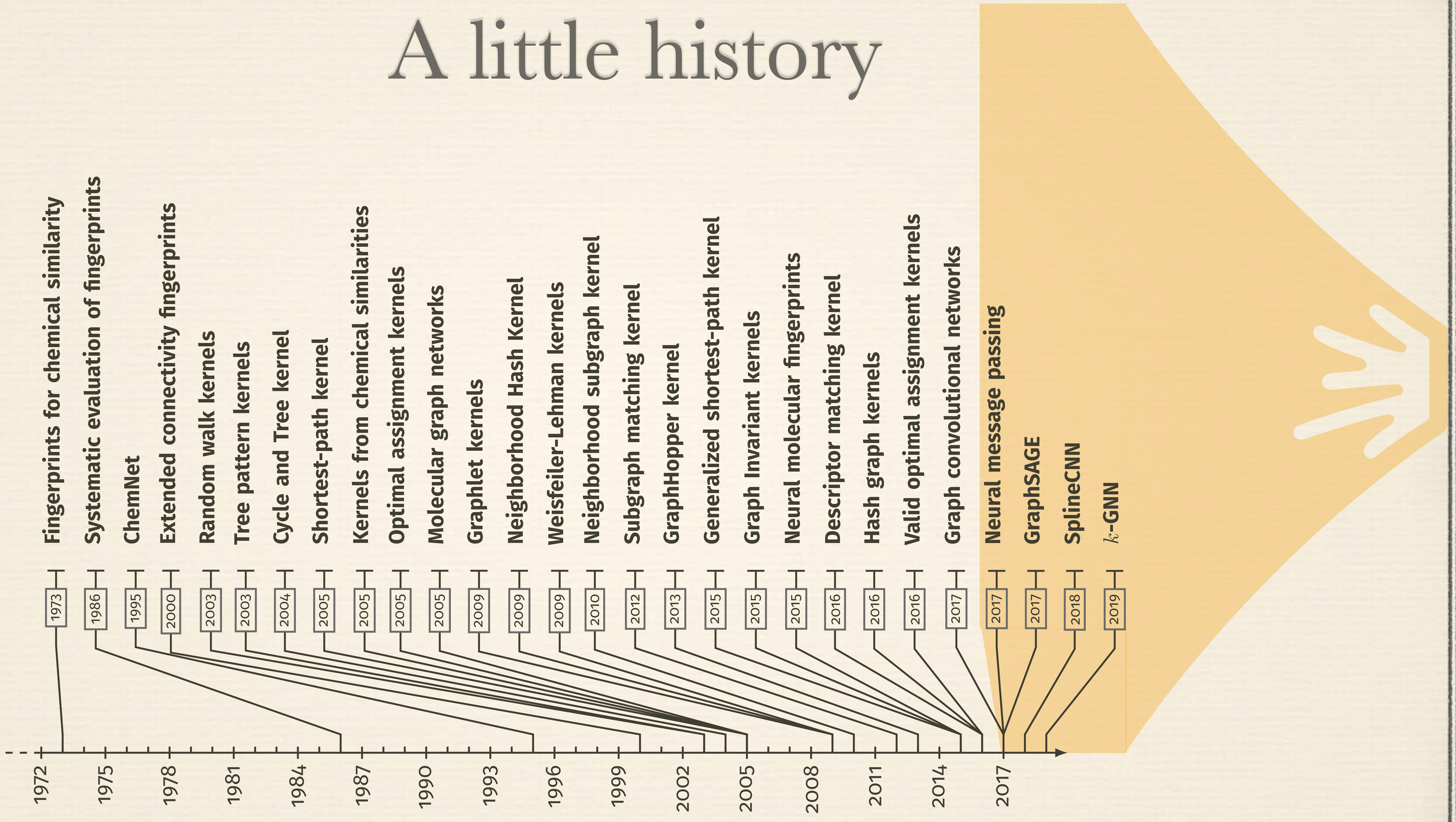
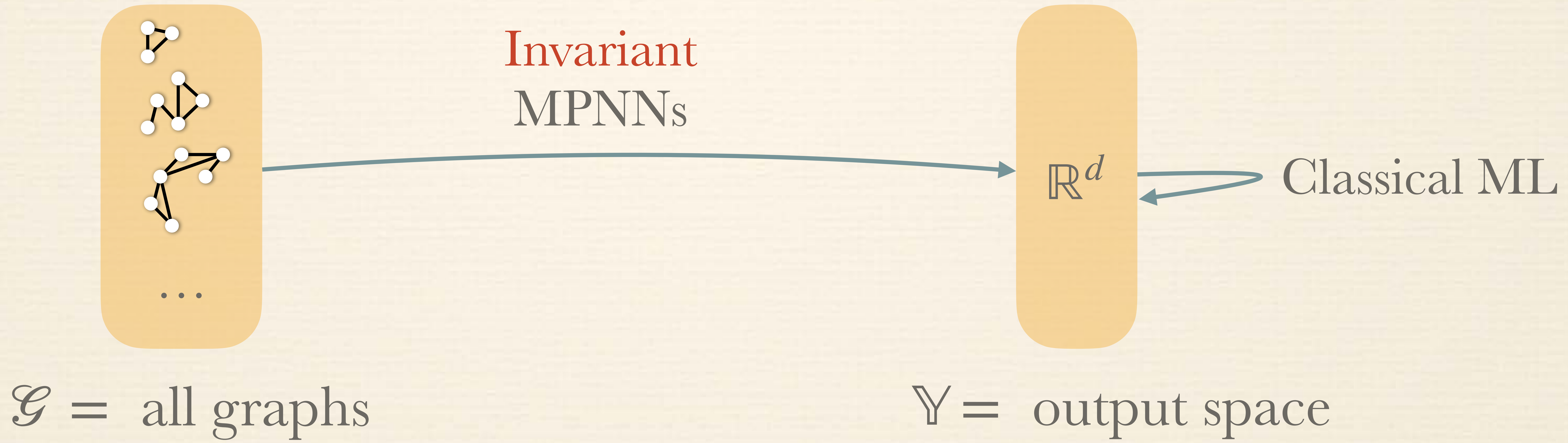


Image: Christopher Morris

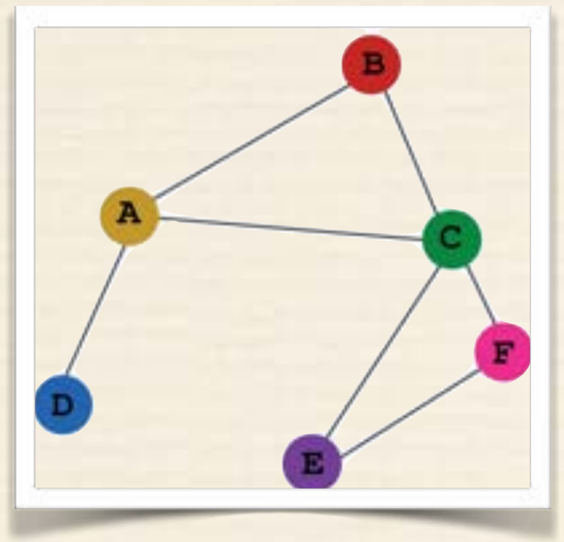
Message passing neural networks

A class of invariant vertex and graph embedding methods

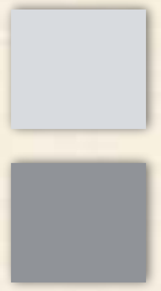


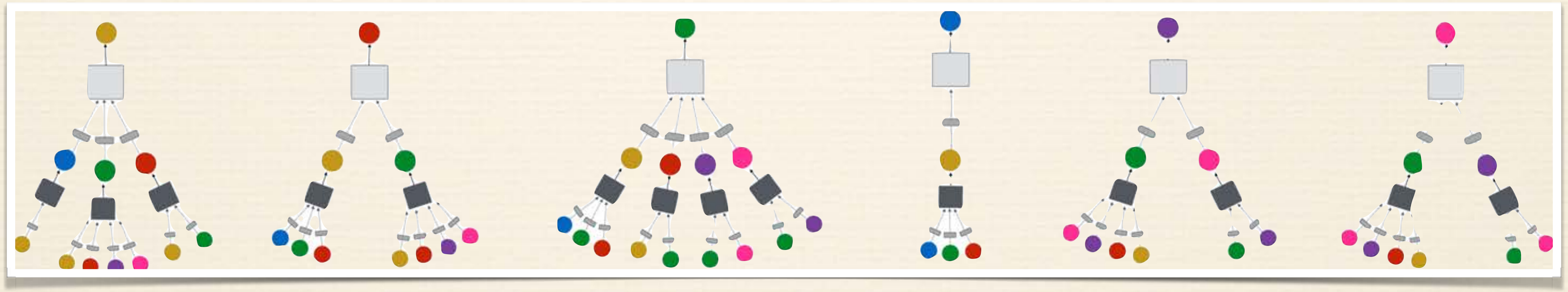
Scarcelli et al.: *The graph neural network model* (2005),
Hamilton et al.: *Inductive representation learning on large graphs* (2017)
Gilmer et al.: *Neural message passing for quantum chemistry* (2017)

Idea behind MPNNs: Neighbour aggregation



Every vertex defines a computation graph


Neural networks



Images: Machine Learning on Graphs, Course by Jure Leskovec

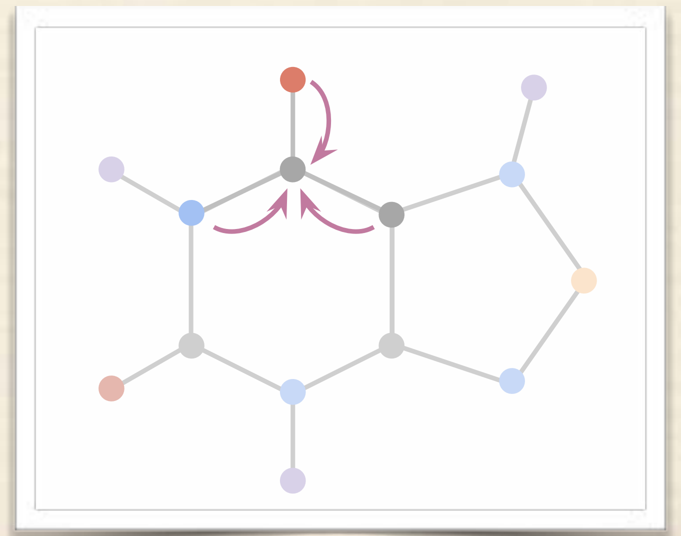
MPNNs: Vertex embedding

$$\xi(G, v) := \xi^{(L)} \circ \xi^{(L-1)} \circ \cdots \circ \xi^{(0)}(G, v)$$

Message Passing Layers

$$\xi^{(0)}(G, v) := \text{Hot-one encoding of label of vertex } v$$

$$\xi^{(t)}(G, v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G, v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G, v), \xi^{(t)}(G, u) \mid u \in N_G(v)\}\}\big)\Big)$$



MPNNs: Vertex embedding

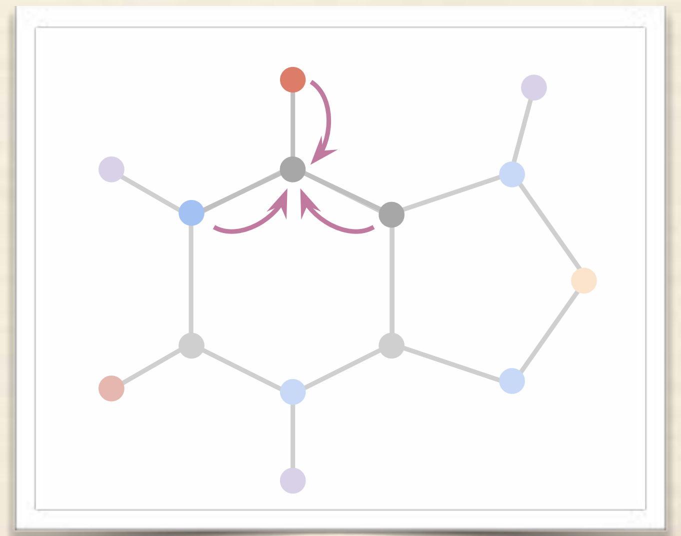
$$\xi(G,v) := \xi^{(L)} \circ \xi^{(L-1)} \circ \cdots \circ \xi^{(0)}(G,v)$$

Message Passing Layers

$$\xi^{(0)}(G,v) := \text{Hot-one encoding of label of vertex } v$$

$$\xi^{(t)}(G,v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G,v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G,v), \xi^{(t)}(G,u) \mid u \in N_G(v)\}\}\big)\Big)$$

Message Passing between v and its neighbours $u \in N_G(v)$



MPNNs: Vertex embedding

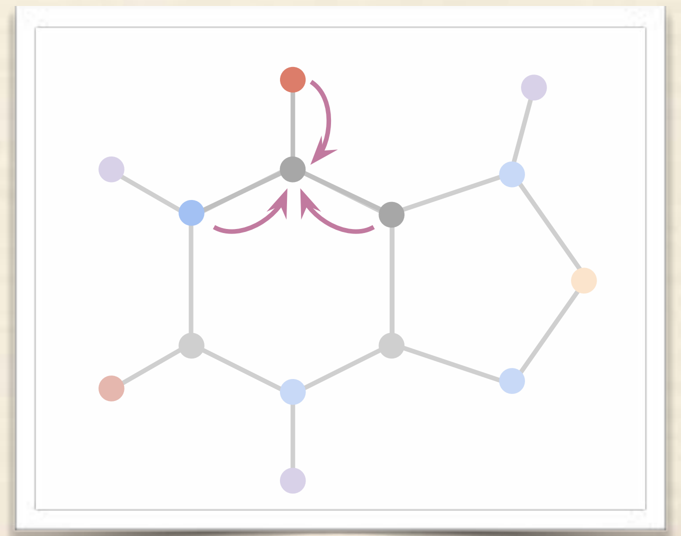
$$\xi(G,v) := \xi^{(L)} \circ \xi^{(L-1)} \circ \dots \circ \xi^{(0)}(G,v)$$

Message Passing Layers

$$\xi^{(0)}(G,v) := \text{Hot-one encoding of label of vertex } v$$

$$\xi^{(t)}(G,v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G,v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G,v), \xi^{(t)}(G,u) \mid u \in N_G(v)\}\}\big)\Big)$$

Message Passing between v and its neighbours $u \in N_G(v)$



Update and aggregate function contain learnable parameters (NNs)

MPNNs: Graph embedding

$$\rho(G) := \rho \circ \xi^{(L)} \circ \xi^{(L-1)} \circ \cdots \circ \xi^{(0)}(G, v)$$

Readout

$$\rho(G) := \mathsf{Readout}\Big(\{\!\{\xi^{(L)}(G, v) \mid v \in V_G\}\!\}\Big)$$

Has learnable parameters

Typical choices for update, aggregate and readout: Multilayer Perceptrons

MPNNs: Graph embedding

$$\rho(G) := \rho \circ \xi^{(L)} \circ \xi^{(L-1)} \circ \cdots \circ \xi^{(0)}(G, v)$$

Readout

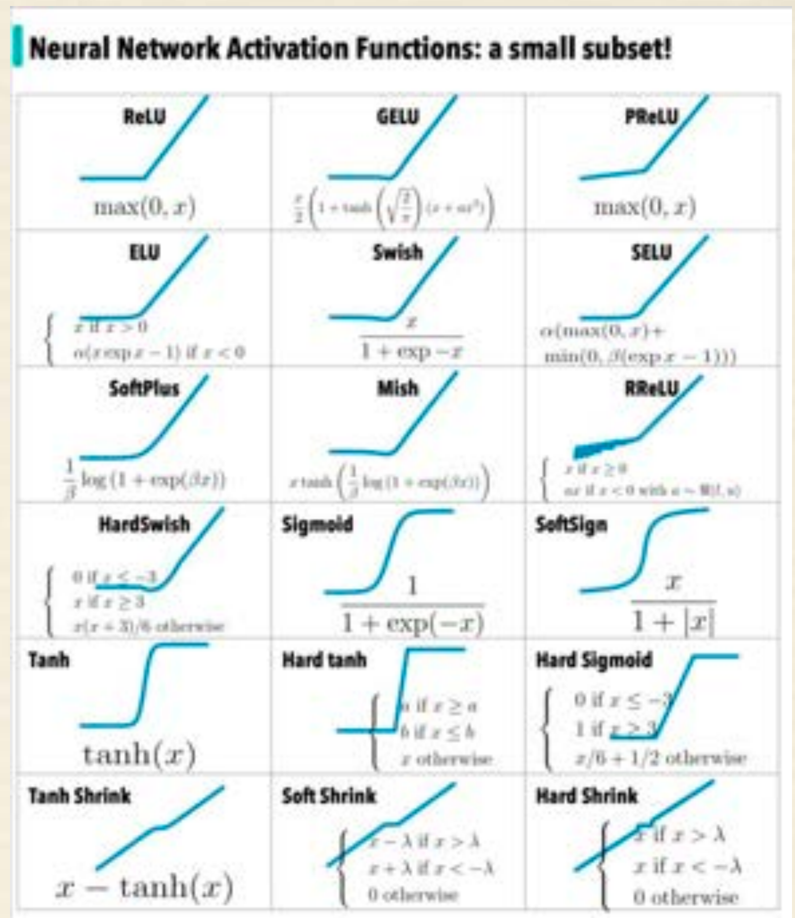
$$\rho(G) := \mathsf{Readout}\Big(\{\{\xi^{(L)}(G, v) \mid v \in V_G\}\}\Big)$$

Has learnable parameters

Aggregation over <u>all</u> vertices

<u>Typical choices</u> for update, aggregate and readout: Multilayer Perceptrons

MPNN example: GNN 101



- Non-linear activation function σ (ReLU, sign, sigmoid, …)
- $\mathbf{F}_{v\bullet}^{(t)} \in \mathbb{R}^d$ denotes embedding of vertex v
- Weight matrices $\mathbf{W}_1^{(t)} \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_2^{(t)} \in \mathbb{R}^{d \times d}$ and bias vector $\mathbf{b} \in \mathbb{R}^{1 \times d}$

$\mathbf{F}_{v\bullet}^{(0)} := L_G(v)$ ← Embedding vertex labels

$$\mathbf{F}_{v\bullet}^{(t)} := \sigma\left(\mathbf{F}_{v\bullet}^{(t-1)}\mathbf{W}_1^{(t)} + \sum_{u \in N_G(v)} \mathbf{F}_{u\bullet}^{(t-1)}\mathbf{W}_2^{(t)} + \mathbf{b}^{(t)}\right)$$

Aggregation over neighbours

Matrix form $\mathbf{F}^{(t)} := \sigma\left(\mathbf{F}^{(t-1)}\mathbf{W}_1^{(t)} + \mathbf{A}\mathbf{F}^{(t-1)}\mathbf{W}_2^{(t)} + \mathbf{B}^{(t)}\right)$

Image: TheAiEdge.io

GNN 101: Graph embedding

- Weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$ and and bias vector $\mathbf{b} \in \mathbb{R}^{1 \times d}$

$$\mathbf{F}^{(t)} := \sigma\left(\boxed{\sum_{v \in V_G}} \mathbf{F}^{(L)}\mathbf{W} + \mathbf{b}\right)$$

Aggregation over all vertices

GNN 101: Graph embedding

- Weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$ and and bias vector $\mathbf{b} \in \mathbb{R}^{1 \times d}$

$$\mathbf{F}^{(t)} := \sigma\left(\sum_{v \in V_G} \mathbf{F}^{(L)}\mathbf{W} + \mathbf{b}\right)$$

Aggregation over all vertices

ERM: Find best parameters $\mathbf{W}_1^{(1)}, \ldots, \mathbf{W}_1^{(L)}, \mathbf{W}_2^{(1)} \ldots, \mathbf{W}_2^{((L)}, \mathbf{W}, \mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(L)}, \mathbf{b}$

Two more examples of MPNNs

- Graph Isomorphism Networks (GIN)

$$\mathbf{F}_{v\bullet}^{(t)} := \mathsf{MLP}^{(t)}\left((1+\epsilon^{(t)})\mathbf{F}_{v\bullet}^{(t-1)} + \sum_{u\in N_G(v)} \mathbf{F}_{u\bullet}^{(t-1)}\right)$$

- Graph Convolution Network (GCN)

$$\mathbf{F}_{v\bullet}^{(t)} := \mathsf{MLP}^{(t)}\left(\frac{1}{\sqrt{|N_G(v)+1}} \sum_{u\in N_G(v)\cup\{u\}} \frac{1}{\sqrt{|N_G(u)+1}}\mathbf{F}_{u\bullet}^{(t-1)}\right)$$

GIN: Xu et al.: *How powerful are graph neural networks?* (2019)
GCN: Kipf and Welling: *Semi-supervised classification with graph convolutional networks* (2017)

MPNNs: Expressive power

What is $\rho(\text{MPNNs})$?

MPNNs: Expressive power

What is $\rho(\text{MPNNs})$?

Recall: All pairs of graphs (G, H) such that all MPNNs return same graph embedding on both graphs.

MPNNs: Expressive power

What is ρ(MPNNs)?

Recall: All pairs of graphs (G, H) such that all MPNNs return same graph embedding on both graphs.

Understanding ρ(MPNNs) translates in understanding power of GNN 101, GCNs, GINs, ….

MPNNs: Expressive power

What is ρ(MPNNs)?

Recall: All pairs of graphs (G, H) such that all MPNNs return same graph embedding on both graphs.

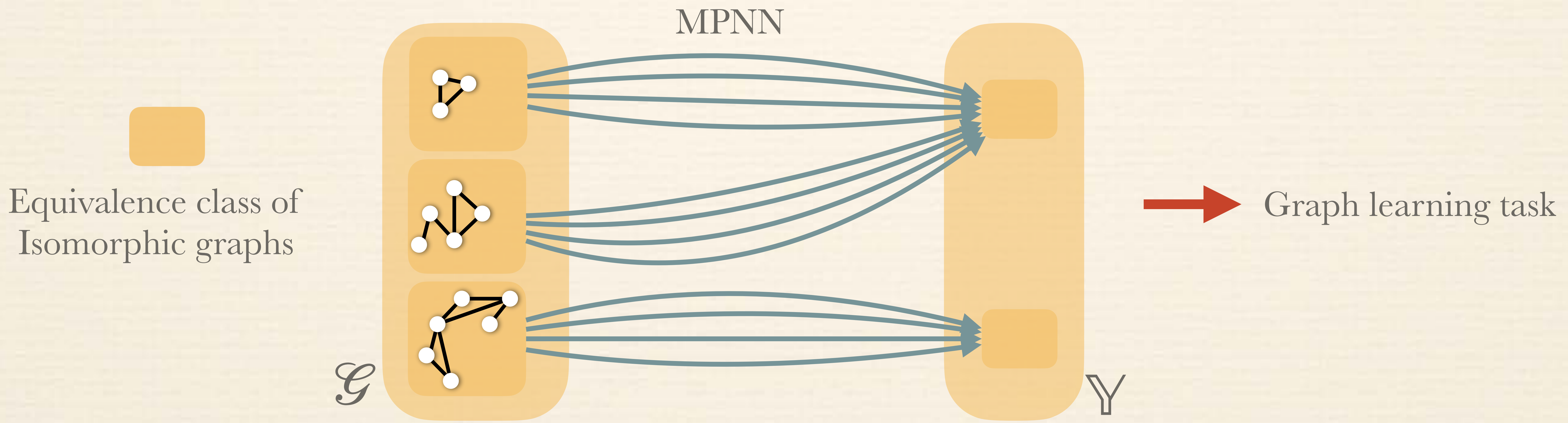
Understanding ρ(MPNNs) translates in understanding power of GNN 101, GCNs, GINs, ….

A short detour to graph isomorphism testing

MPNNs and isomorphic graphs

- Because of invariance: MPNNs embed isomorphic graphs in the same way. That is, if $G \cong H \Rightarrow (G, H) \in \rho(\mathrm{MPNN})$

- Can MPNNs embed non-isomorphic graphs differently?



The graph isomorphism problem

Given two graph $G = (V_G, E_G, L_G)$ and $H = (V_H, E_H, L_H)$: are they isomorphic? Or is $G \cong H$?

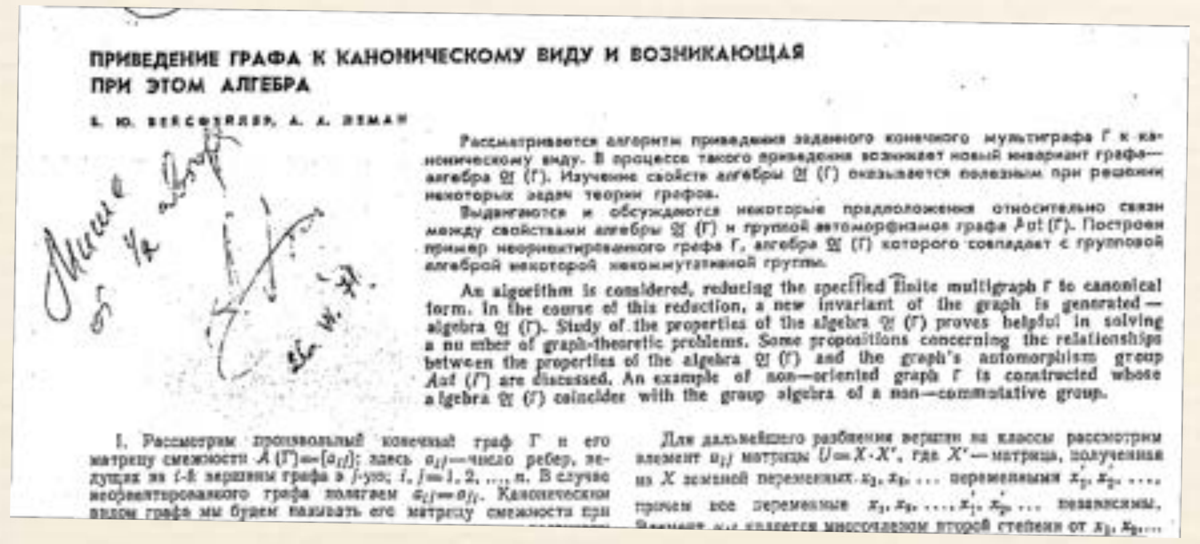
- Does there exist a graph isomorphism $\pi : V_G \to V_H$?
- Theory: computational complexity is open.
- Quasi-polynomial algoritm $n^{\log(n)^{\mathcal{O}(1)}}$ by László Babai (2016).
- Practice: mostly solvable very fast.

L. Babai: *Graph isomorphism in quasipolynomial time* (2016)

One-sided test: Colour refinement

Apply heuristic on G and H: If Heuristic say "no" then $G \ncong H$, otherwise we do not know.

- Common heuristic is colour refinement
- In paper 1968 by Boris Weisfeiler and Andrei Leman.



ПРИВЕДЕНИЕ ГРАФА К КАНОНИЧЕСКОМУ ВИДУ И ВОЗНИКАЮЩАЯ ПРИ ЭТОМ АЛГЕБРА

Б. Ю. ВЕЙСФЕЙЛЕР, А. А. ЛЕМАН

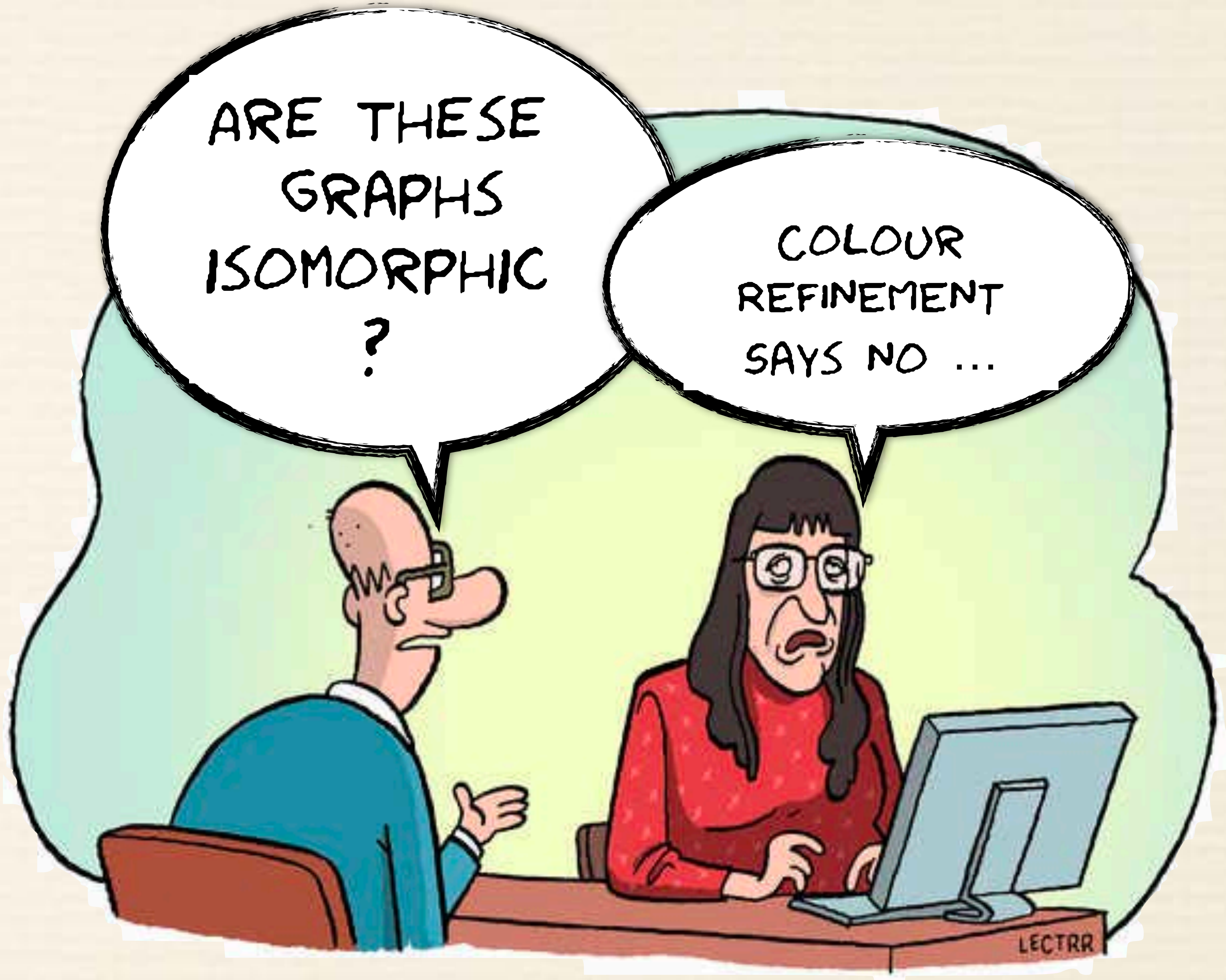
Рассматривается алгоритм приведения заданного конечного мультиграфа Г к каноническому виду. В процессе такого приведения возникает новый инвариант графа — алгебра $\mathfrak{A}$ (Г). Изучение свойств алгебры $\mathfrak{A}$ (Г) оказывается полезным при решении некоторых задач теории графов.

Выдвигаются и обсуждаются некоторые предположения относительно связи между свойствами алгебры $\mathfrak{A}$ (Г) и группой автоморфизмов графа Aut (Г). Построен пример неориентированного графа Г, алгебра $\mathfrak{A}$ (Г) которого совпадает с групповой алгеброй некоторой некоммутативной группы.

An algorithm is considered, reducing the specified finite multigraph Γ to canonical form. In the course of this reduction, a new invariant of the graph is generated — algebra $\mathfrak{A}$ (Γ). Study of the properties of the algebra $\mathfrak{A}$ (Γ) proves helpful in solving a number of graph-theoretic problems. Some propositions concerning the relationships between the properties of the algebra $\mathfrak{A}$ (Γ) and the graph's automorphism group Aut (Γ) are discussed. An example of non-oriented graph Γ is constructed whose algebra $\mathfrak{A}$ (Γ) coincides with the group algebra of a non-commutative group.

1. Рассмотрим произвольный конечный граф Г и его матрицу смежности $A(Г)=[a_{ij}]$; здесь a_{ij} — число ребер, ведущих из i-й вершины графа в j-ую; $i, j = 1, 2, \ldots, n$. В случае неориентированного графа полагаем $a_{ij} = a_{ji}$. Каноническим видом графа мы будем называть его матрицу смежности при

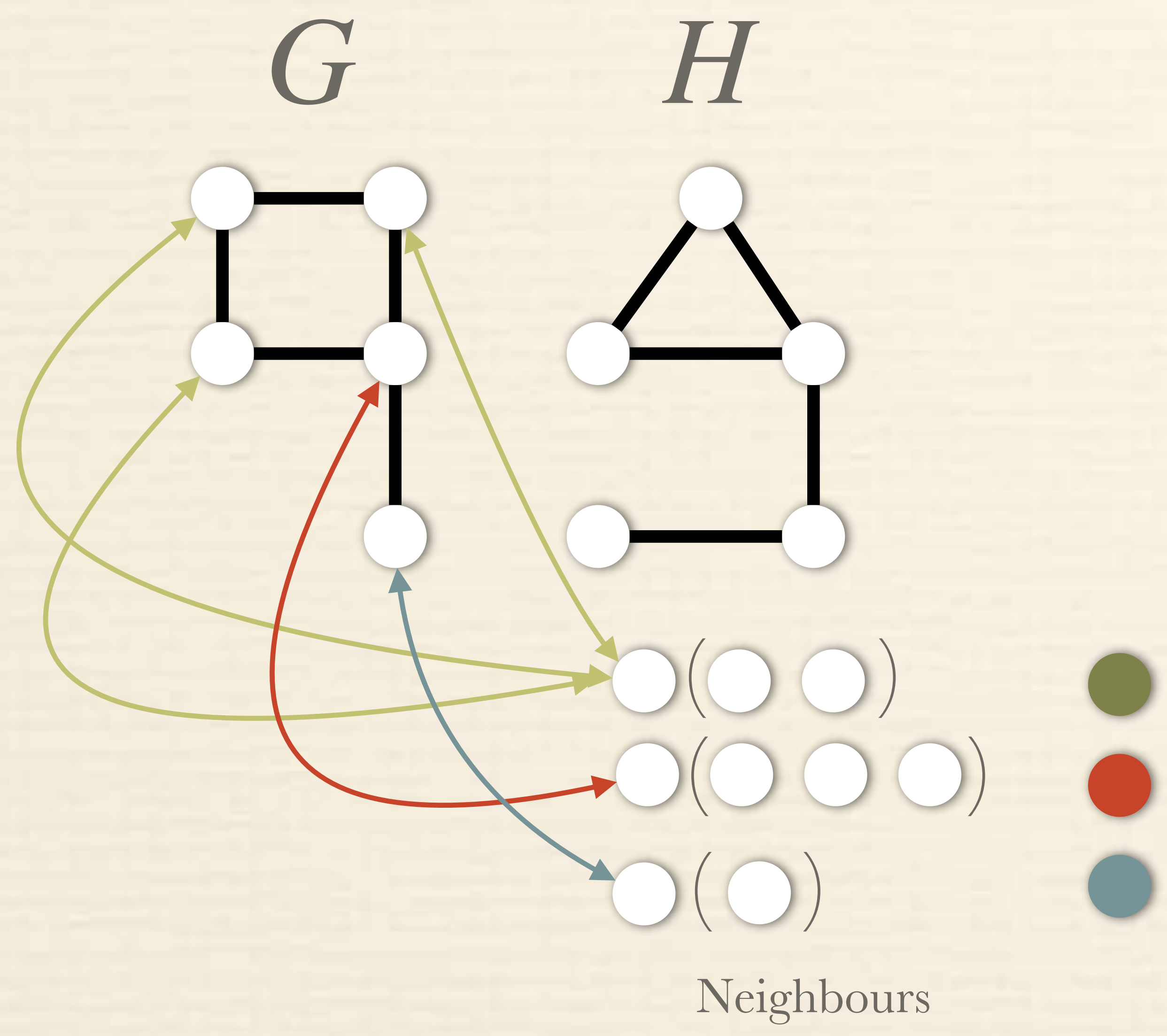
Для дальнейшего разбиения вершин на классы рассмотрим элемент u_{ij} матрицы $U = X \cdot X'$, где X' — матрица, полученная из X заменой переменных $x_1, x_2, \ldots$ переменными $x'_1, x'_2, \ldots$, причем все переменные $x_1, x_2, \ldots, x'_1, x'_2, \ldots$ независимы.



B. Weisfeiler and A. Leman. *The reduction of a graph to canonical form and the algebra which appears therein (1968)*

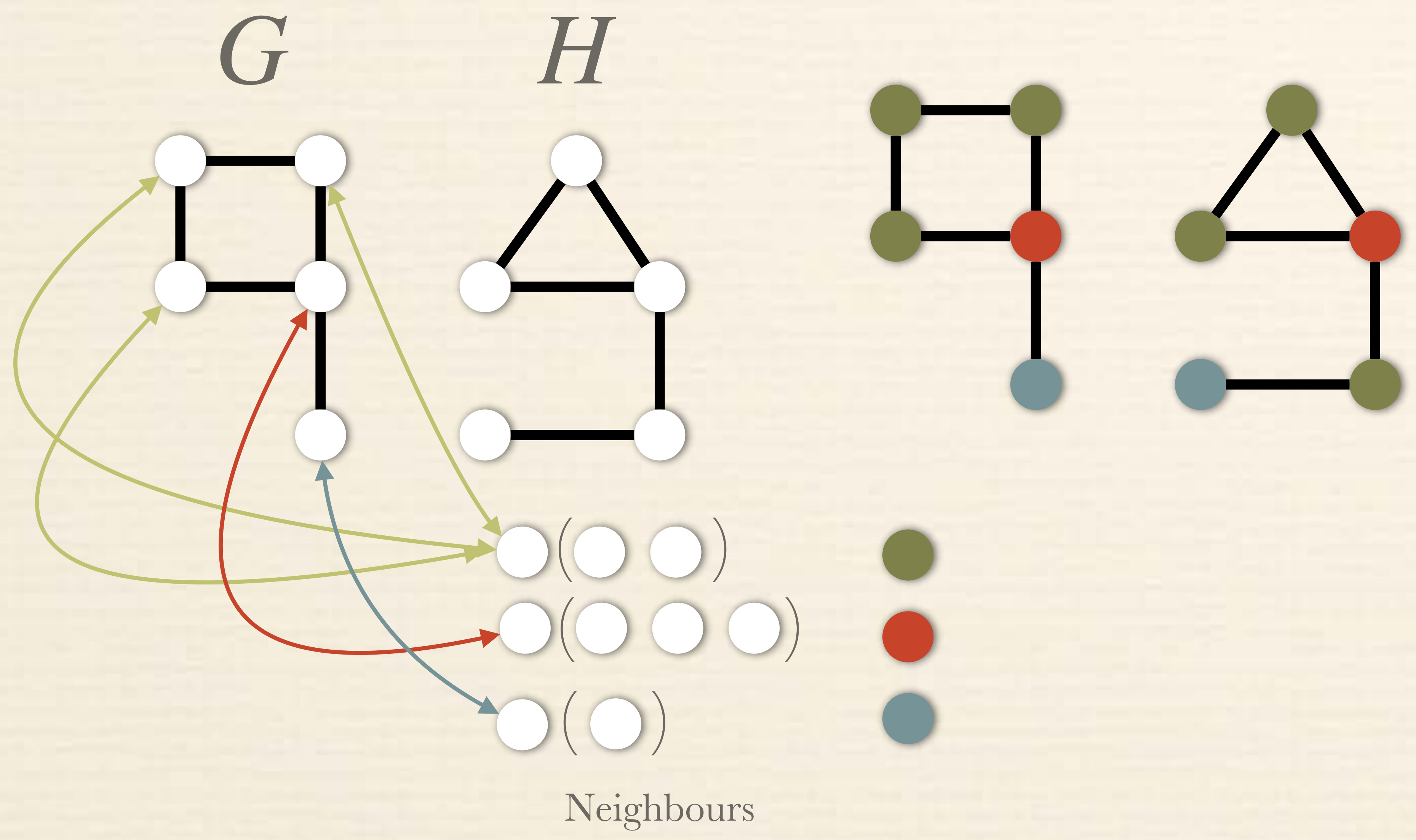
Colour refinement

- Initial: All vertices have their original colour (label)
- Iteration: Separation of identically coloured vertices based on colour histograms of neighbours.
- Two graphs are non-isomorphic if they have different colour histograms.



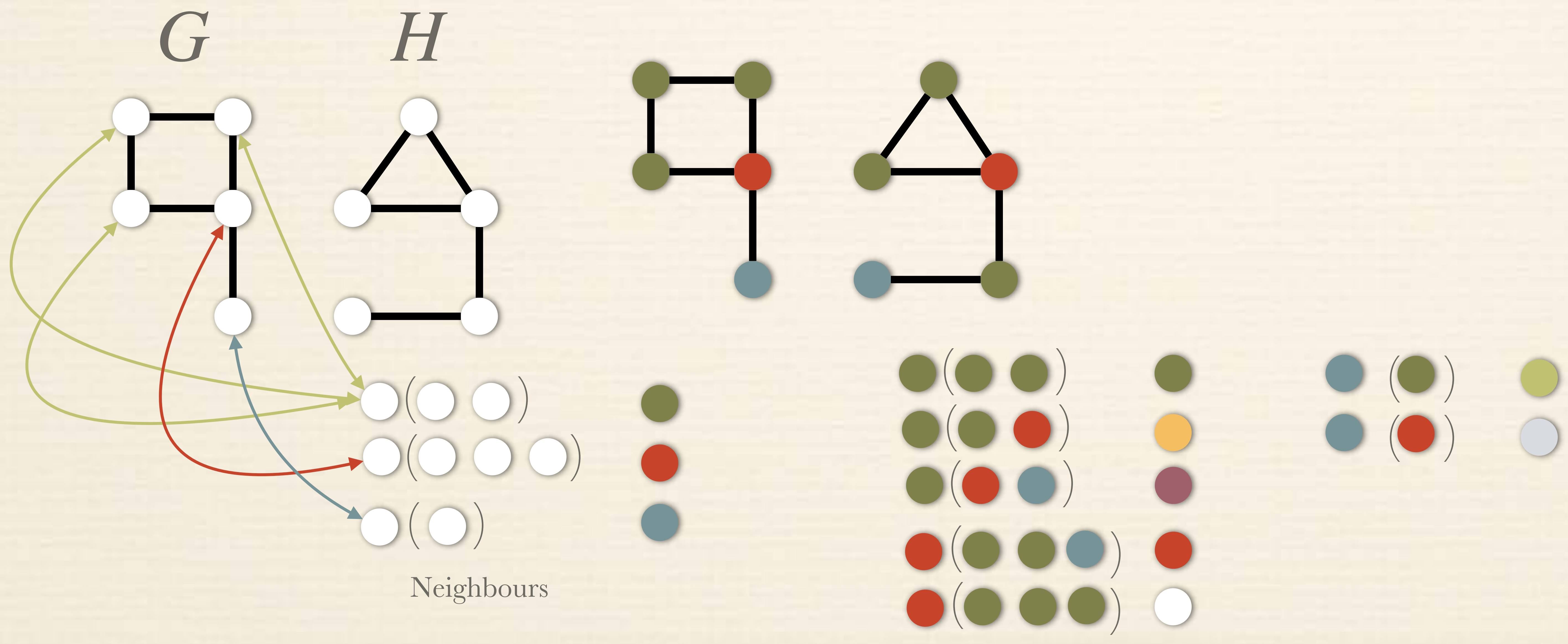
Colour refinement

- Initial: All vertices have their original colour (label)
- Iteration: Separation of identically coloured vertices based on colour histograms of neighbours.
- Two graphs are non-isomorphic if they have different colour histograms.



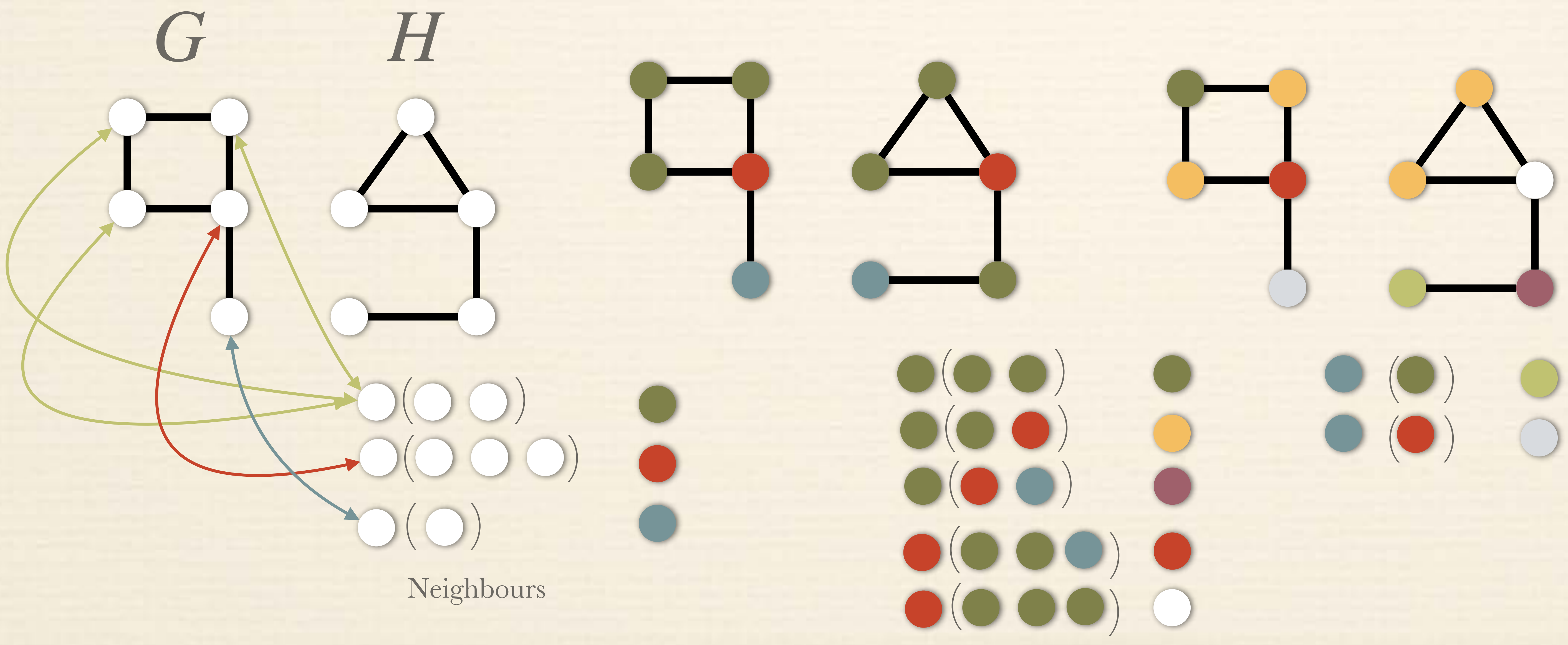
Colour refinement

- Initial: All vertices have their original colour (label)
- Iteration: Separation of identically coloured vertices based on colour histograms of neighbours.
- Two graphs are non-isomorphic if they have different colour histograms.



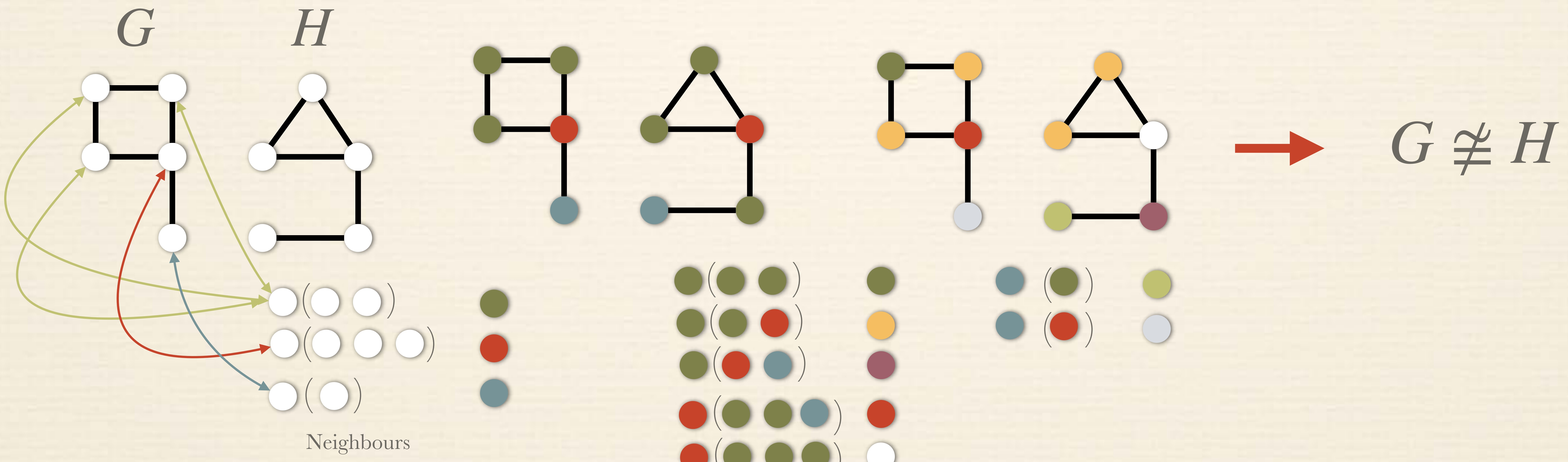
Colour refinement

- Initial: All vertices have their original colour (label)
- Iteration: Separation of identically coloured vertices based on colour histograms of neighbours.
- Two graphs are non-isomorphic if they have different colour histograms.



Colour refinement

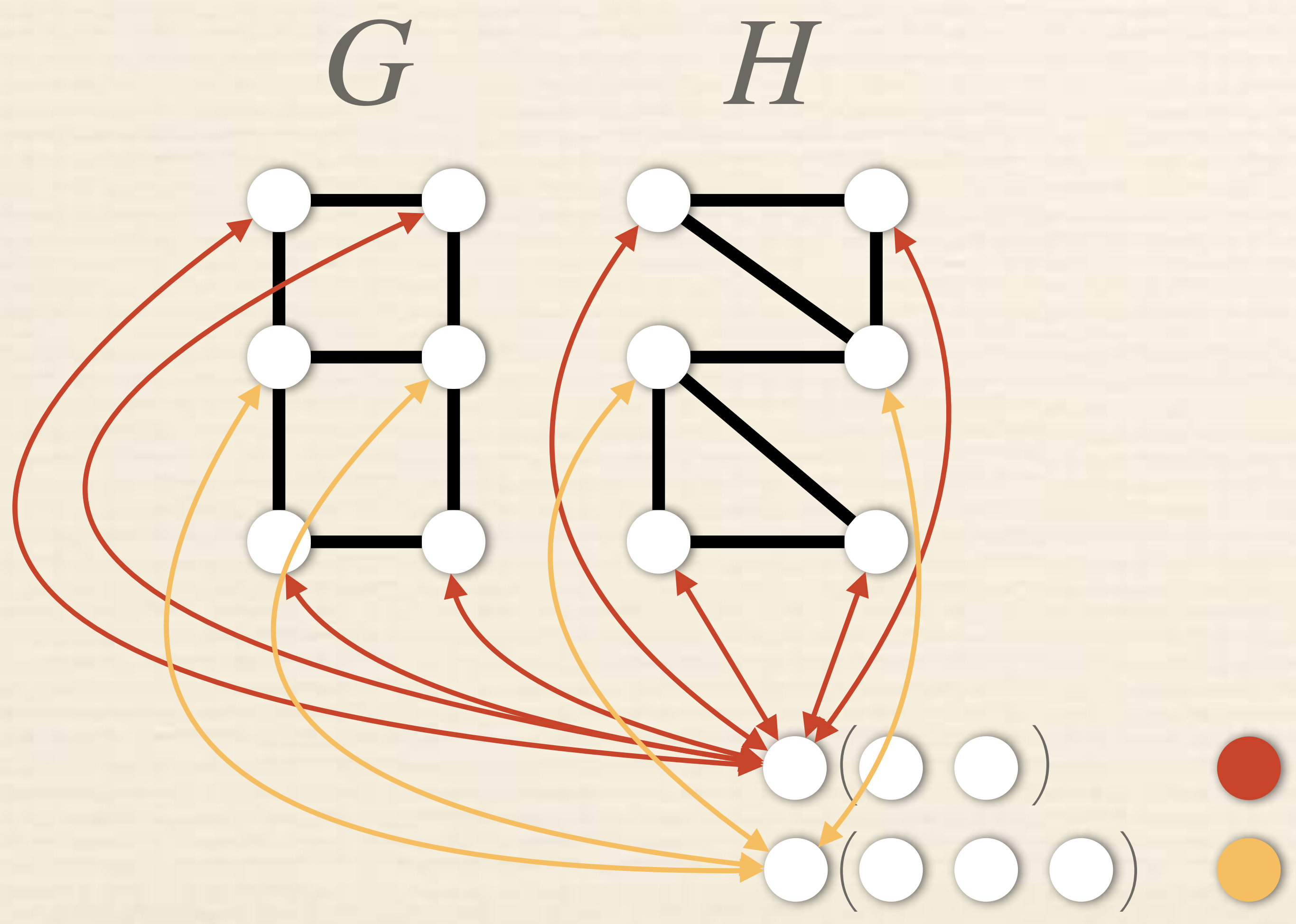
- Initial: All vertices have their original colour (label)
- Iteration: Separation of identically coloured vertices based on colour histograms of neighbours.
- Two graphs are non-isomorphic if they have different colour histograms.



Colour refinement

- Initial: All vertices have their original colour (label)
- Iteration: Separation of identically coloured vertices based on colour histograms of neighbours.
- Two graphs are non-isomorphic if they have different colour histograms.

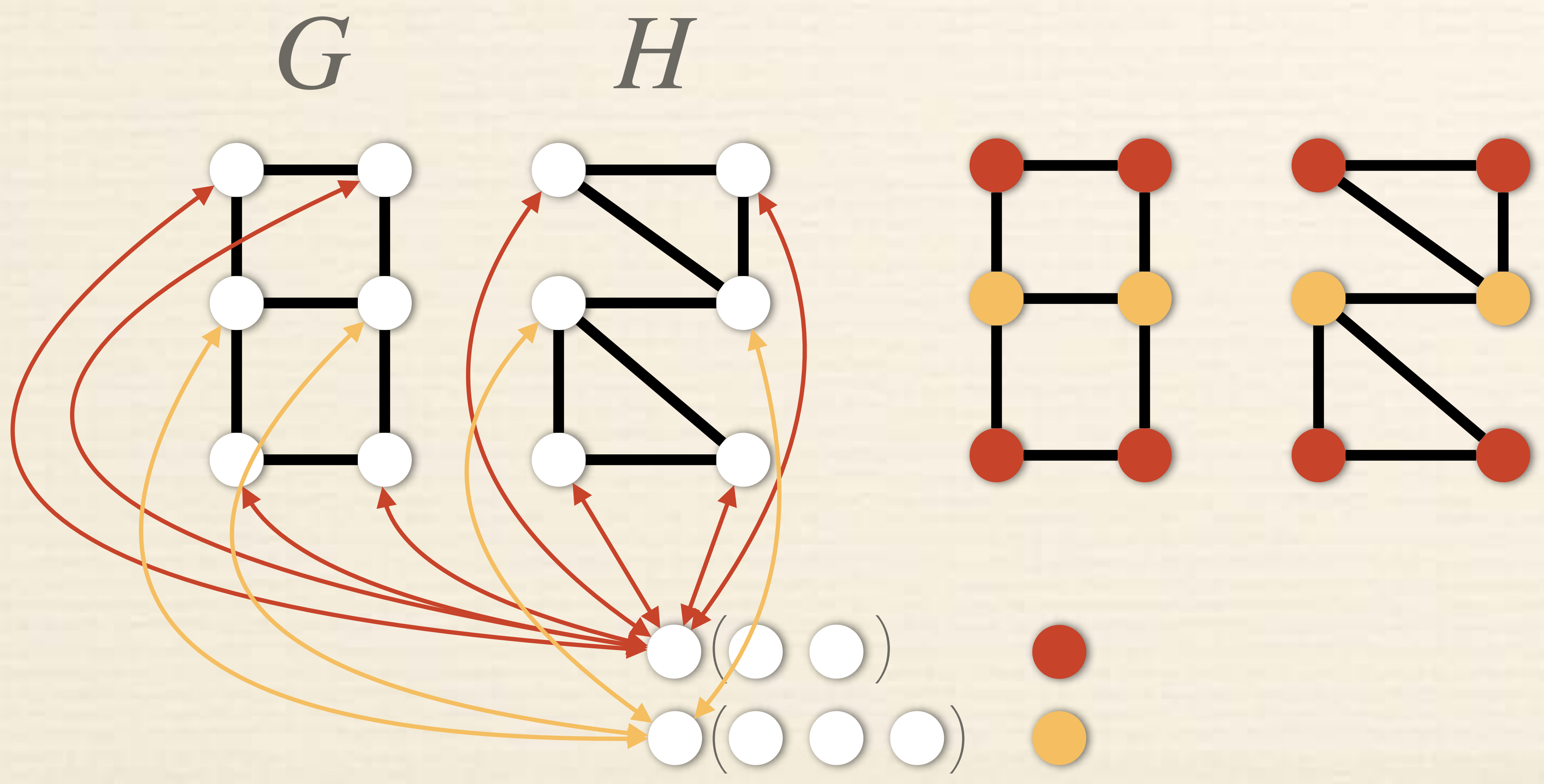
Stops when colour partition does not change (max n iterations)



Colour refinement

- Initial: All vertices have their original colour (label)
- Iteration: Separation of identically coloured vertices based on colour histograms of neighbours.
- Two graphs are non-isomorphic if they have different colour histograms.

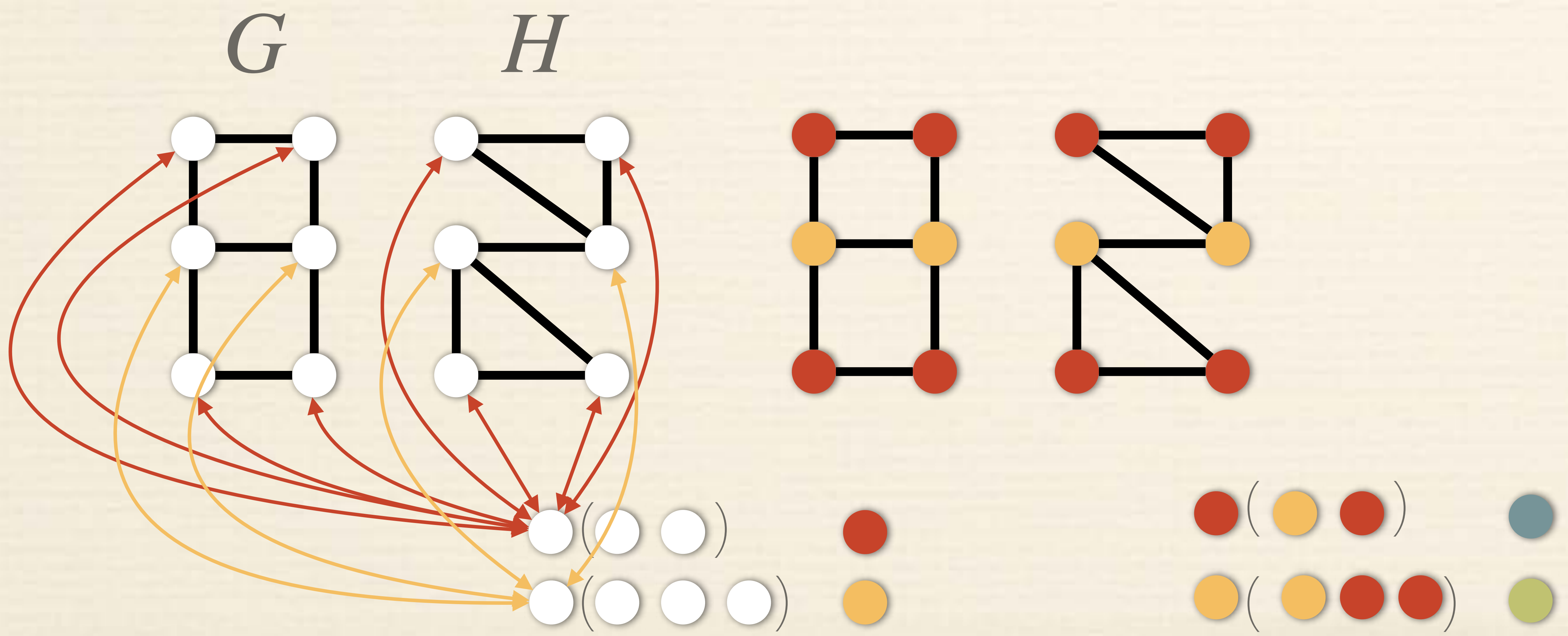
Stops when colour partition does not change (max n iterations)



Colour refinement

- Initial: All vertices have their original colour (label)
- Iteration: Separation of identically coloured vertices based on colour histograms of neighbours.
- Two graphs are non-isomorphic if they have different colour histograms.

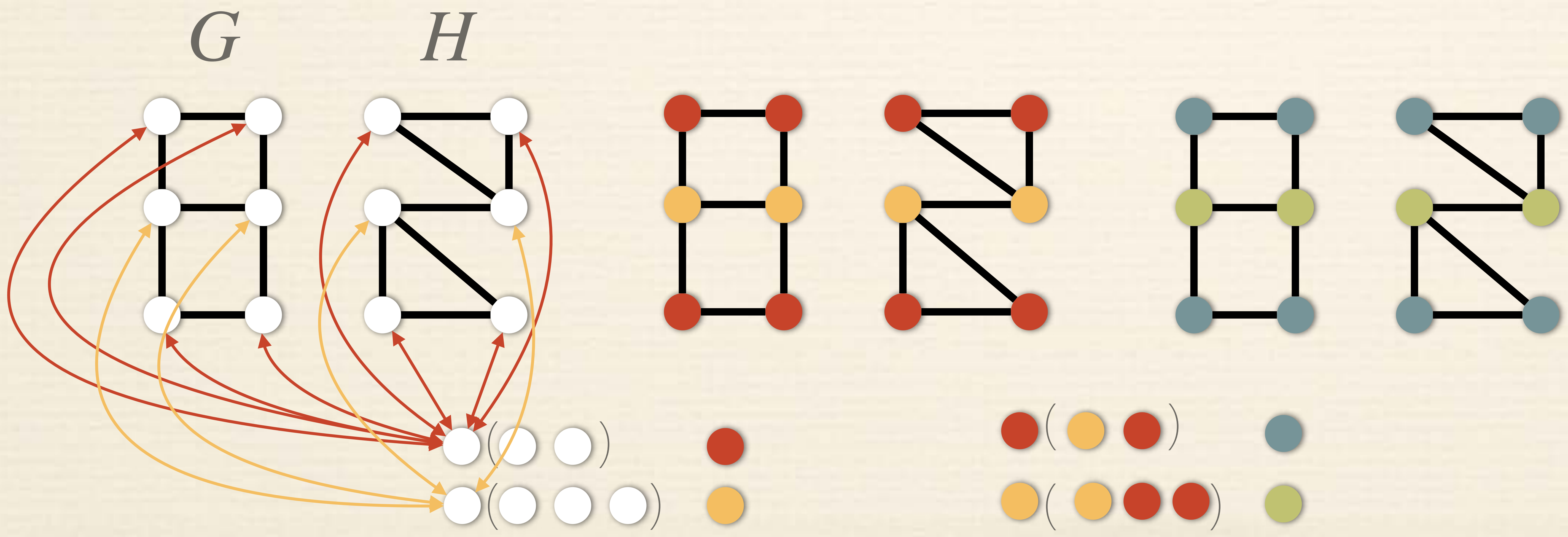
Stops when colour partition does not change (max n iterations)



Colour refinement

- Initial: All vertices have their original colour (label)
- Iteration: Separation of identically coloured vertices based on colour histograms of neighbours.
- Two graphs are non-isomorphic if they have different colour histograms.

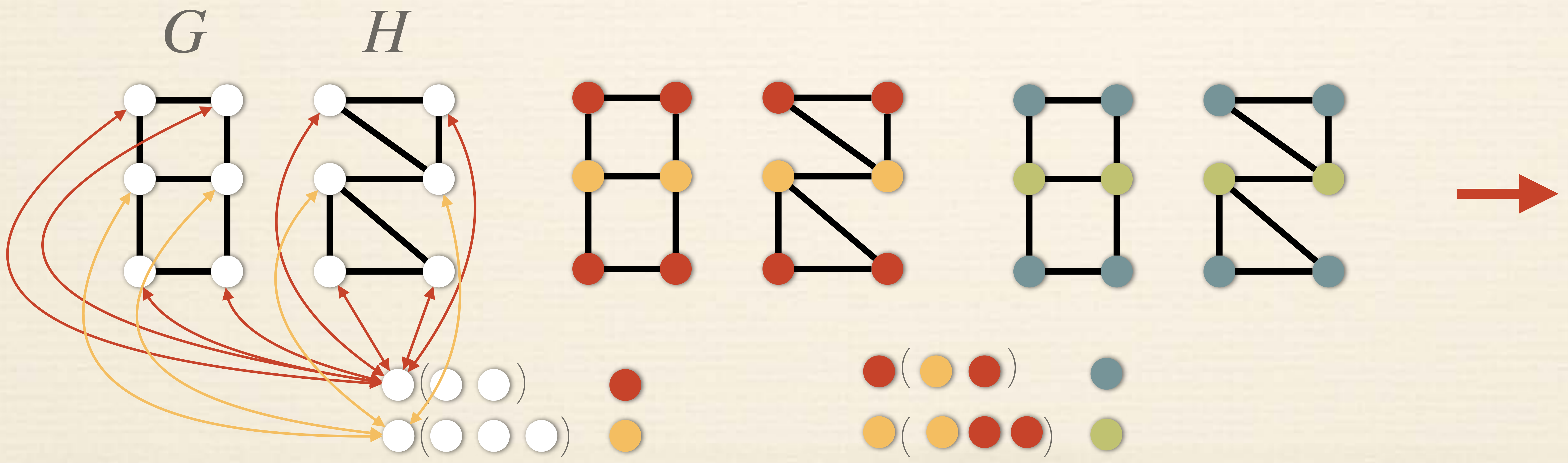
Stops when colour partition does not change (max n iterations)



Colour refinement

- Initial: All vertices have their original colour (label)
- Iteration: Separation of identically coloured vertices based on colour histograms of neighbours.
- Two graphs are non-isomorphic if they have different colour histograms.

Stops when colour partition does not change (max n iterations)



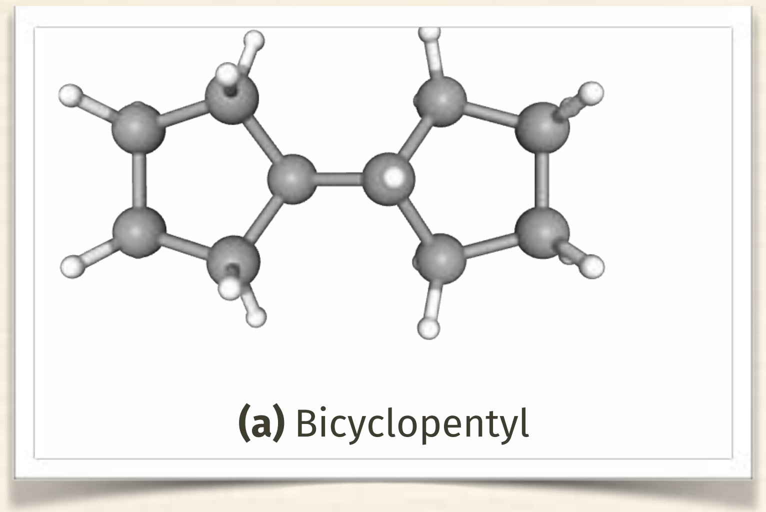
$\rightarrow$ $G \cong H$?

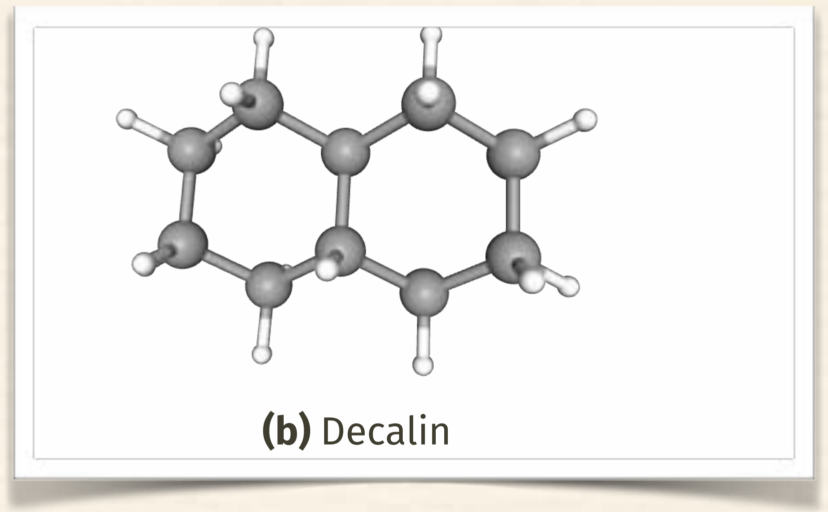
Color refinement

- Extensively studied in the theoretical computer science community
- Many different characterisations of when two graphs have the same colour histograms (equivalent for colour refinement).
- Successful on random graphs with high probability
- Weak expressive power

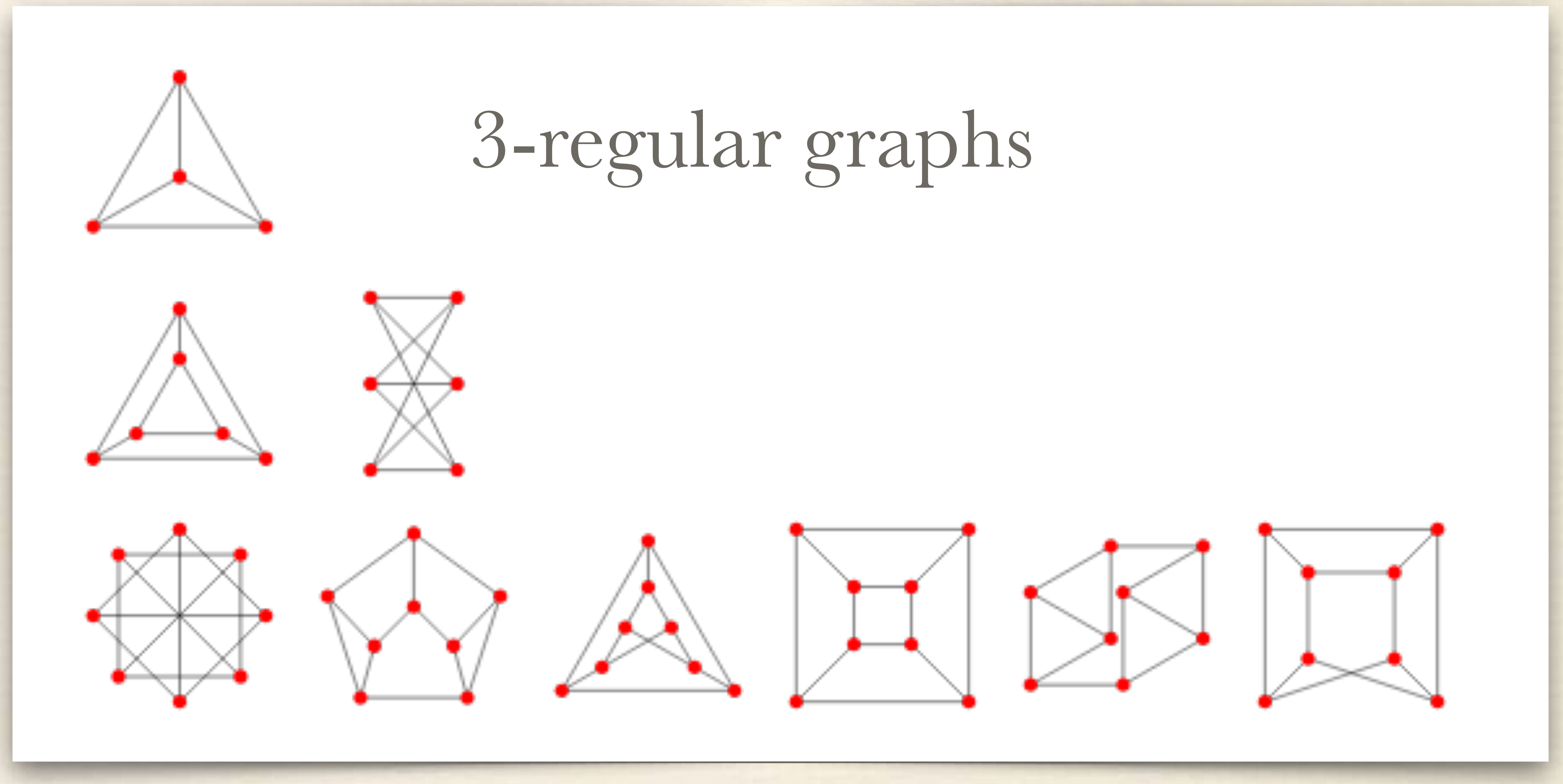
L. Babai and L. Kucera. *Canonical labelling of graphs in linear average time* (1979)
Cai et al.: *An optimal lower bound on the number of variables for graph identifications.* (1992)
Arvind et al.: *On the power of color refinement* (2015)
M. Grohe: *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory* (2017)
Arvind et al.: *On WL invariance: Subgraph Counts and related properties* (2019)
M. Grohe. *The logic of graph neural networks* (2021)

ρ(colour refinement)



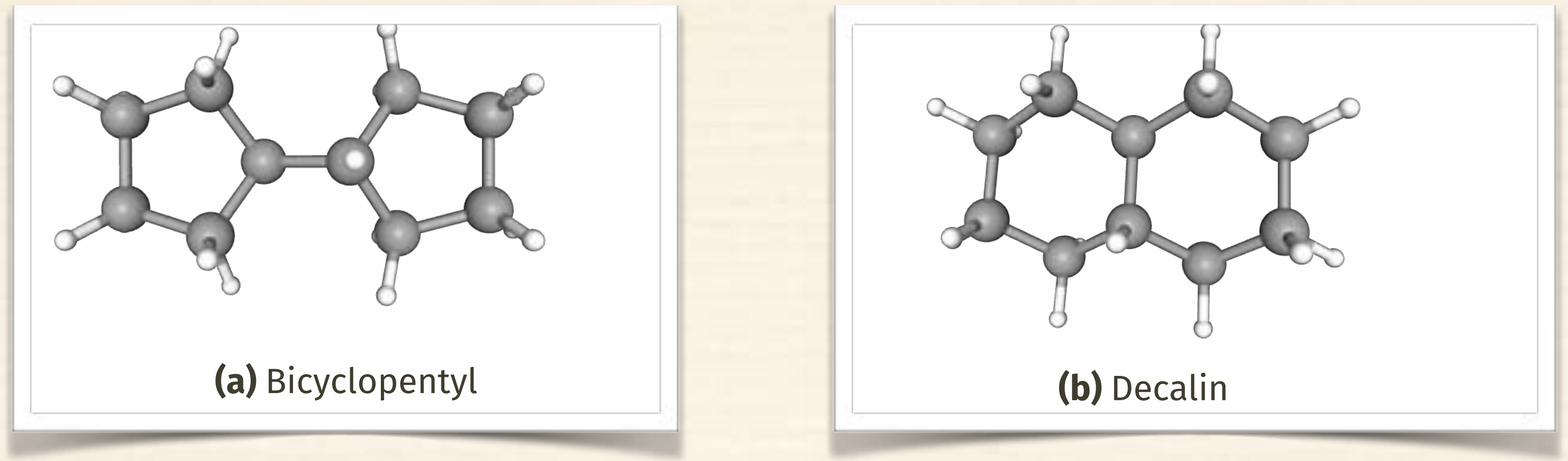


- Cannot distinguish d-regular graphs
- Cannot count cycles (triangles)
- Only tree information

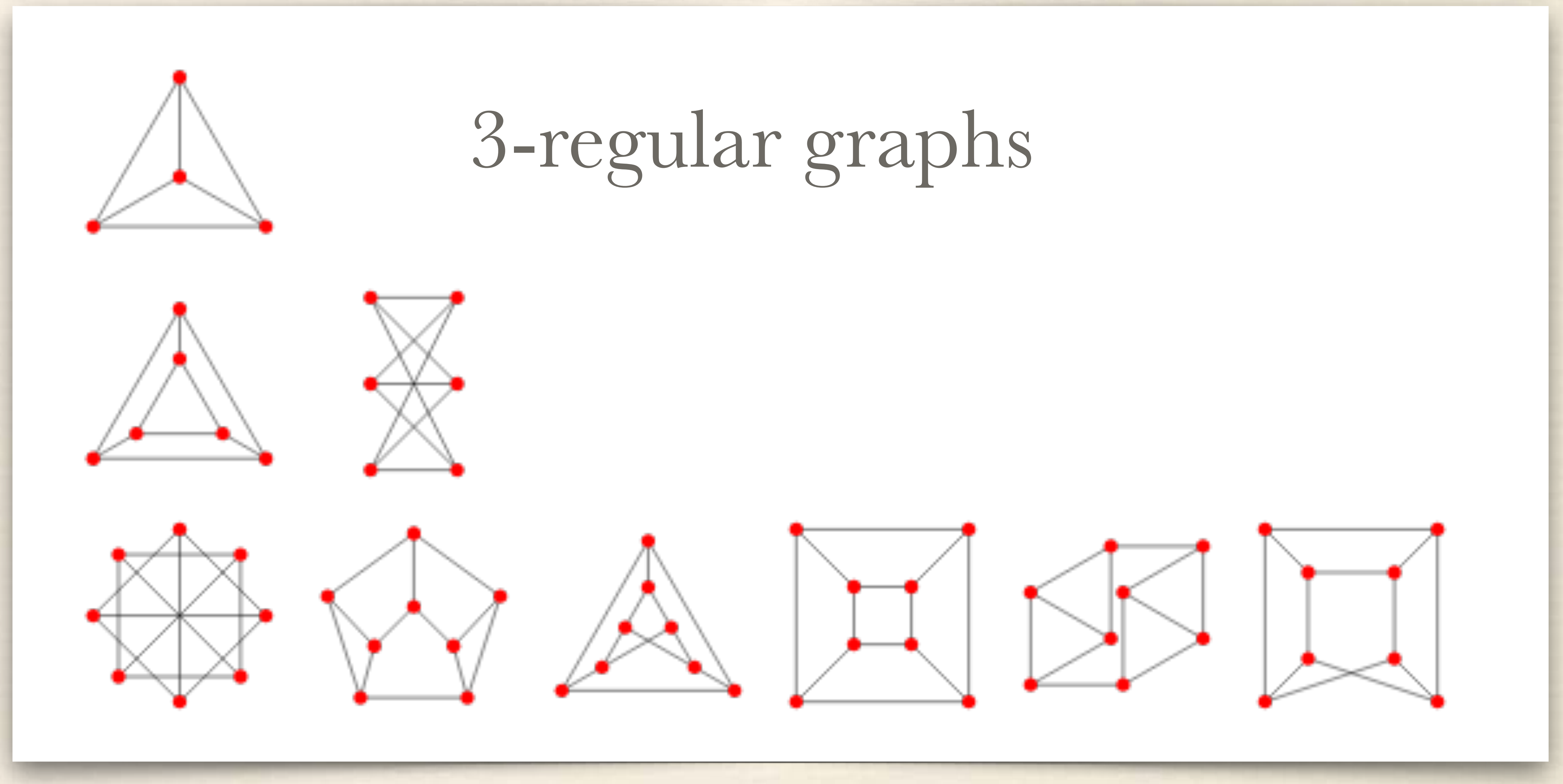


Arvind et al.: *On the power of color refinement* (2015)
Images: Wolfram MathWorld, Christopher Morris

ρ(colour refinement)



- Cannot distinguish d-regular graphs
- Cannot count cycles (triangles)
- Only tree information



Back to MPNNs

Arvind et al.: *On the power of color refinement* (2015)
Images: Wolfram MathWorld, Christopher Morris

MPNNs & Colour refinement

Theorem (Xu et al. 2019, Morris et al. 2019)

If colour refinement cannot tell two graphs apart then neither can any MPNN!

Xu et al.: *How powerful are graph neural networks?* (2019)
Morris et al: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019)

MPNNs & Colour refinement

Theorem (Xu et al. 2019, Morris et al. 2019)

If colour refinement cannot tell two graphs apart then neither can any MPNN!

MPNNs

$\xi^{(0)}(G, v)$:= Hot-one encoding of label of vertex v

$$\xi^{(t)}(G, v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G, v), \mathsf{Agg}^{(t)}(\{\{\xi^{(t-1)}(G, v), \xi^{(t)}(G, u) \mid u \in N_G(v)\}\})\Big)$$

$$\rho(G) := \mathsf{Readout}\Big(\{\{\xi^{(L)}(G, v) \mid v \in V_G\}\}\Big)$$

Color refinement

$\mathsf{cr}^{(0)}(G, v)$:= Initial label of v

$$\mathsf{cr}^{(t)}(G, v) := \mathsf{Hash}\Big(\mathsf{cr}^{(t-1)}(G, v), \{\{\mathsf{cr}^{(t-1)}(G, u) \mid u \in N_G(v)\}\}\Big)$$

$$\rho(G) := \{\{\mathsf{cr}(G, v) \mid v \in V_G\}\}$$

Xu et al.: *How powerful are graph neural networks?* (2019)
Morris et al: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019)

MPNNs & Colour refinement

Theorem (Xu et al. 2019, Morris et al. 2019)

If colour refinement cannot tell two graphs apart then neither can any MPNN!

MPNNs

$\xi^{(0)}(G,v) :=$ Hot-one encoding of label of vertex v

$$\xi^{(t)}(G,v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G,v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G,v), \xi^{(t)}(G,u) \mid u \in N_G(v)\}\}\big)\Big)$$

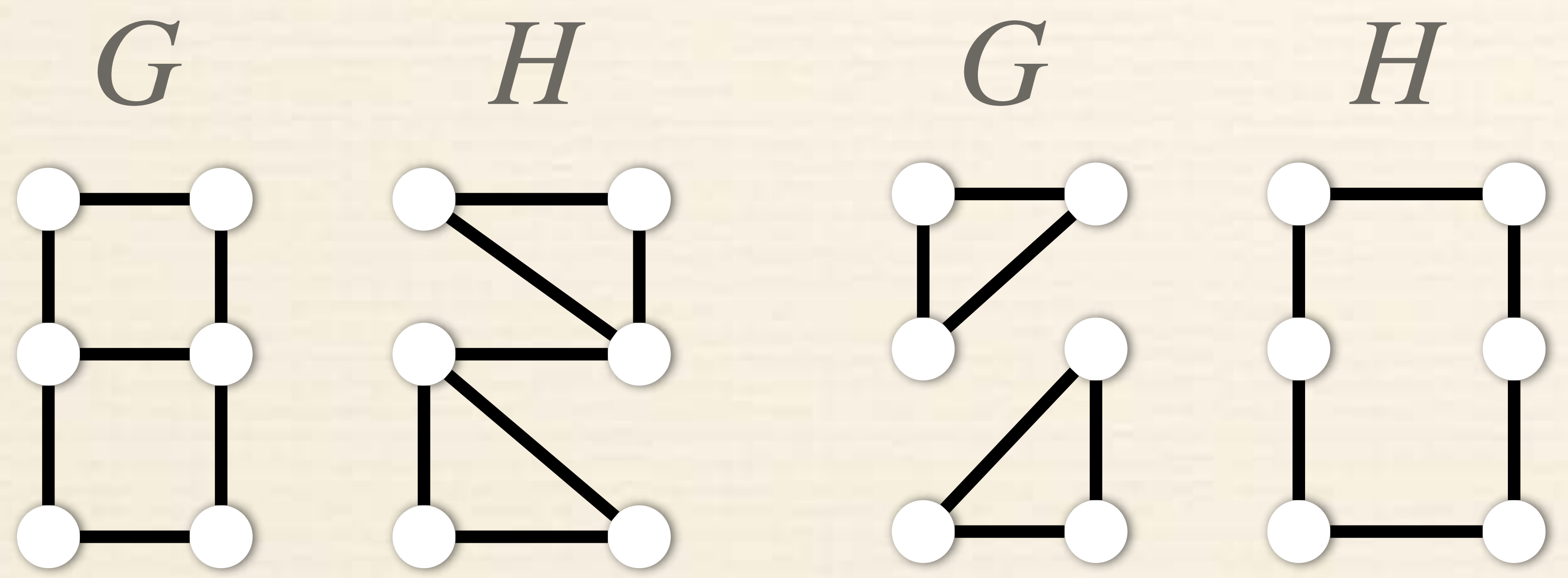
$$\rho(G) := \mathsf{Readout}\Big(\{\{\xi^{(L)}(G,v) \mid v \in V_G\}\}\Big)$$

Color refinement

$\mathsf{cr}^{(0)}(G,v) :=$ Initial label of v

$$\mathsf{cr}^{(t)}(G,v) := \mathsf{Hash}\Big(\mathsf{cr}^{(t-1)}(G,v), \{\{\mathsf{cr}^{(t-1)}(G,u) \mid u \in N_G(v)\}\}\Big)$$

$$\rho(G) := \{\{\mathsf{cr}(G,v) \mid v \in V_G\}\}$$

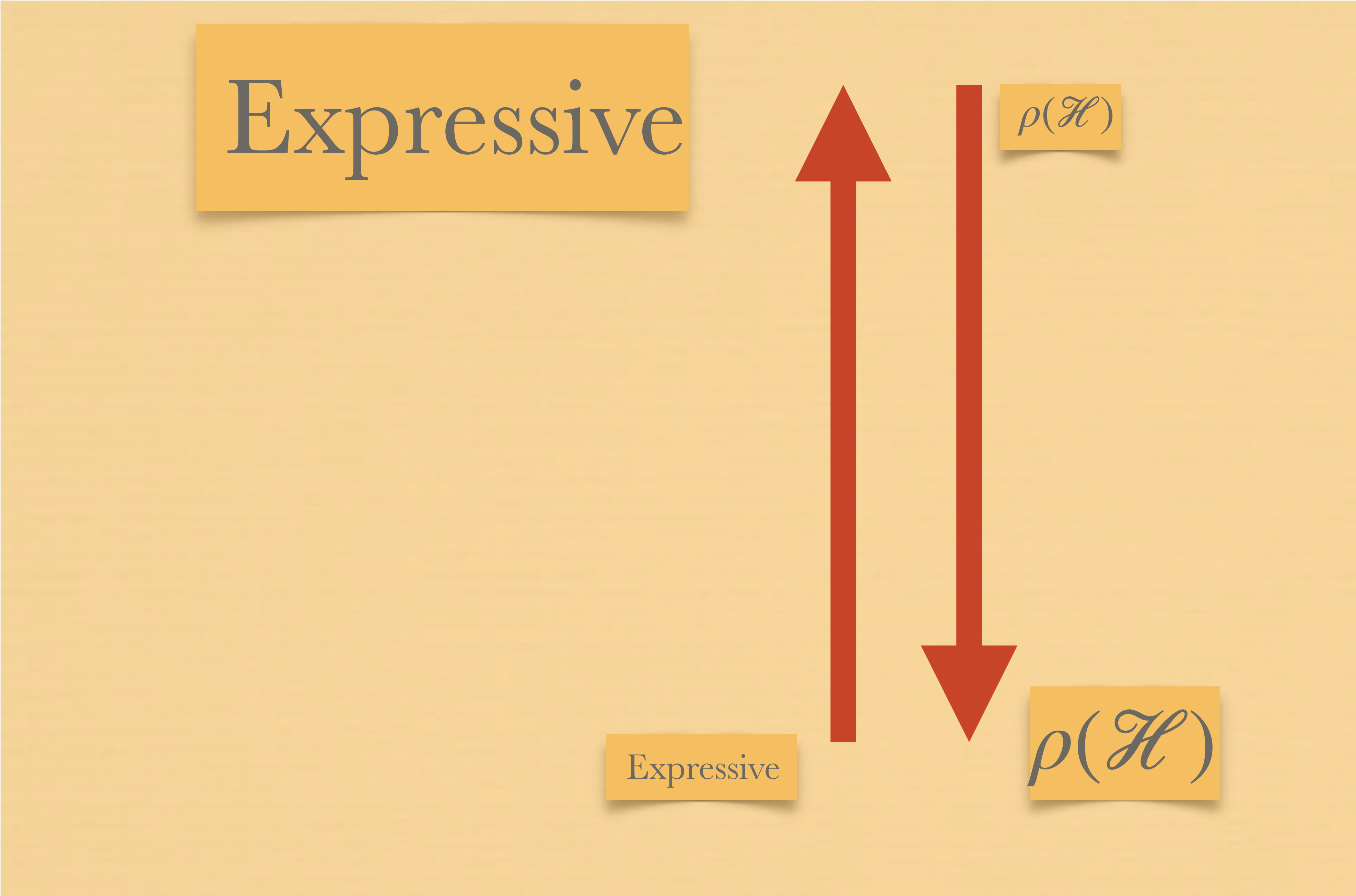


→ No MPNN can separate these graphs

Xu et al.: *How powerful are graph neural networks?* (2019)
Morris et al: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019)

MPNNs & Colour refinement

Recall:



We have just shown: $\rho(\text{colour refinement}) \subseteq \rho(\text{MPNNs})$

Expressive power of MPNNs is upper bounded by colour refinement

Lower bound?

- We have seen that MPNNs cannot separate more graphs than colour refinement.

- Can colour refinement separate more graphs than MPNNs?

Morris et al: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019)

Lower bound?

- We have seen that MPNNs cannot separate more graphs than colour refinement.

- Can colour refinement separate more graphs than MPNNs? No!

Theorem (Morris et al. 2019)

There exists a GNN 101 which can embed G and H differently when colour refinement assigns them different colours

- The class of MPNNs is as powerful (or weak) as colour refinement

Morris et al: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019)

What else can we say?

$\rho(\text{colour refinement}) = \rho(\text{MPNNs})$

What else can we say?

$$\rho(\text{colour refinement}) = \rho(\text{MPNNs})$$

Other - more insightful - characterisations?

What else can we say?

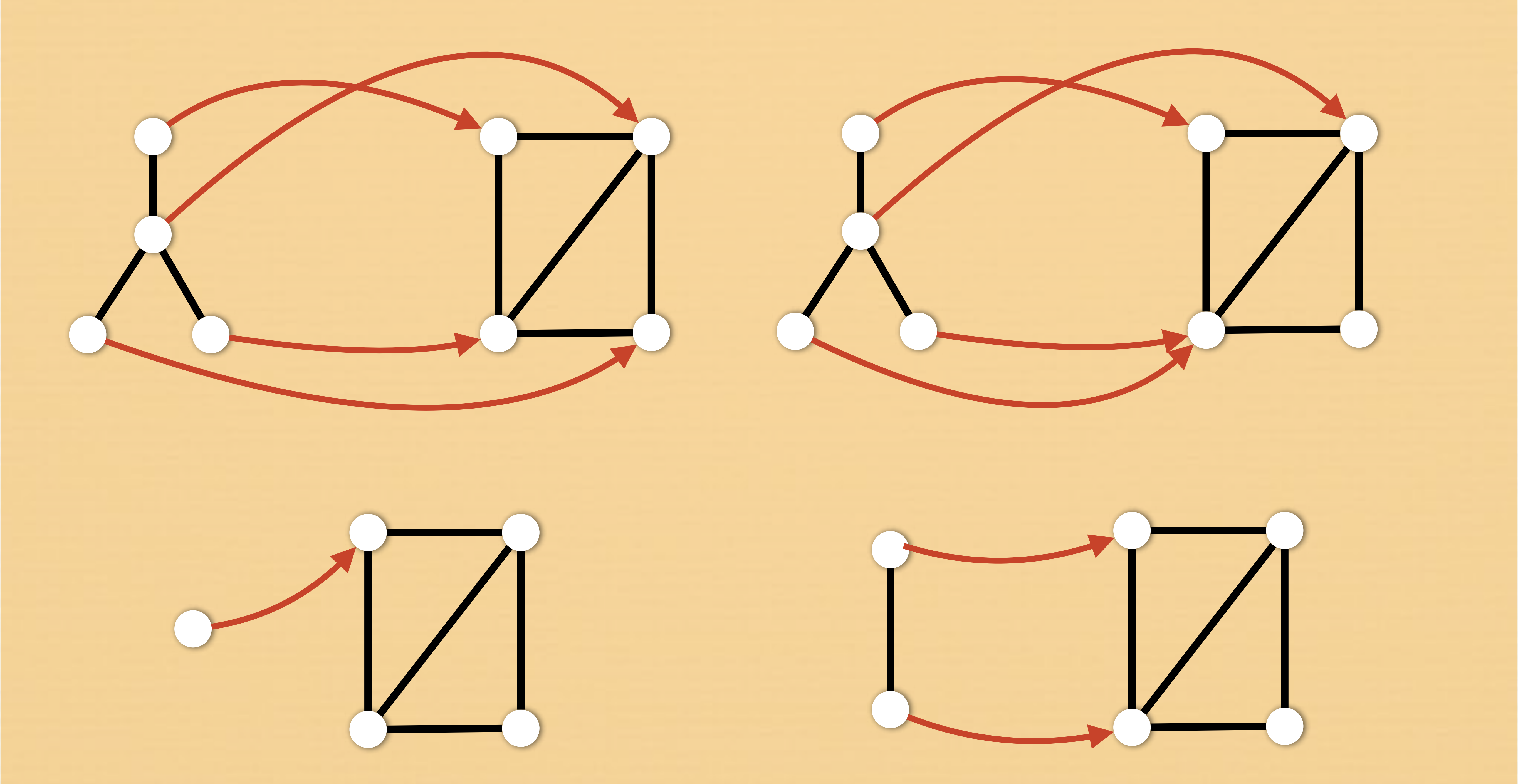
$\rho(\text{colour refinement}) = \rho(\text{MPNNs})$

Other - more insightful - characterisations?

A detour to homomorphism counts

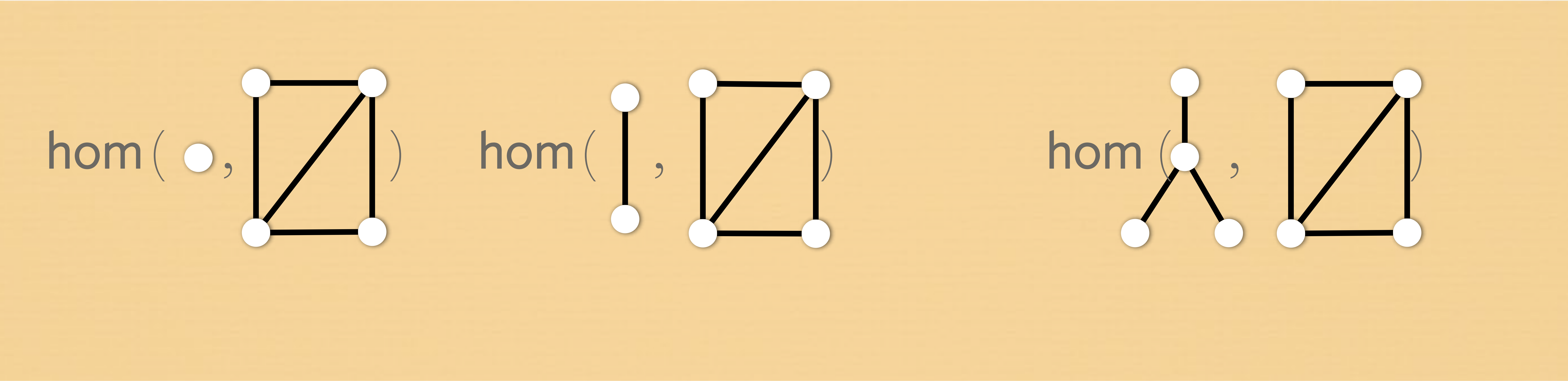
Homomorphisms

- Let $P = (V_P, E_P, L_P)$ and $G = (V_G, E_G, L_G)$ be graphs.
- A function $h : V_P \to V_G$ is a homomorphism if it is edge preserving $(v, w) \in E_p \Rightarrow (h(v), h(w)) \in E_G$ and label preserving.



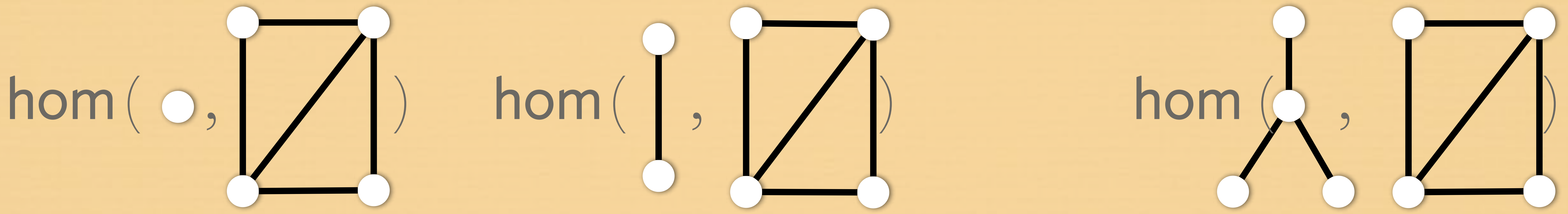
Homomorphism counts

- Define $\mathrm{HOM}(P, G) := \{$ all homomorphisms from P to $G\}$
- Define $\mathrm{hom}(P, G) := |\mathrm{HOM}(P, G)|$.



Homomorphism counts

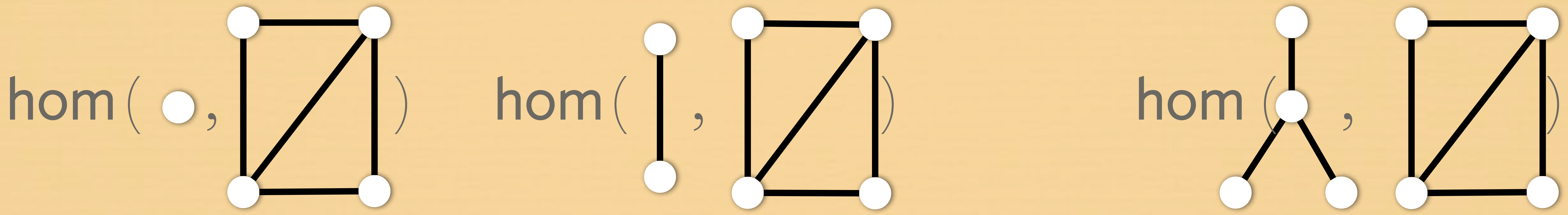
- Define $\mathsf{HOM}(P, G)$:= { all homomorphisms from P to G}
- Define $\mathsf{hom}(P, G)$:= $|\mathsf{HOM}(P, G)|$.



#vertices = 4

Homomorphism counts

- Define $\mathrm{HOM}(P, G)$:= { all homomorphisms from P to G}
- Define $\mathrm{hom}(P, G)$:= $|\mathrm{HOM}(P, G)|$.

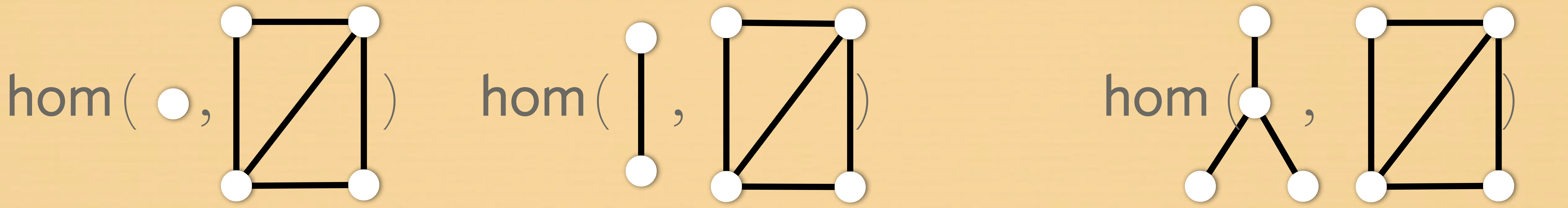


#vertices = 4

2#edges=10

Homomorphism counts

- Define $\mathrm{HOM}(P, G) := \{$ all homomorphisms from P to $G\}$
- Define $\mathrm{hom}(P, G) := |\mathrm{HOM}(P, G)|$.



#vertices = 4

2#edges=10

$70 = 2 \cdot 2^3 + 2 \cdot 3^3$

Homomorphisms

- Weaker notion than subgraph isomorphism (see later)
- Underlies semantics of many graph query languages
- Algebra of homomorphism counts: A rich and active area of research.

Homomorphisms

- Weaker notion than subgraph isomorphism (see later)
- Underlies semantics of many graph query languages
- Algebra of homomorphism counts: A rich and active area of research.

Back to MPNNs

MPNNs and hom counts

Theorem (Dell et al. 2019, Dvorák 2010)

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all *trees* T
if and only if
colour refinement cannot distinguish G from H.

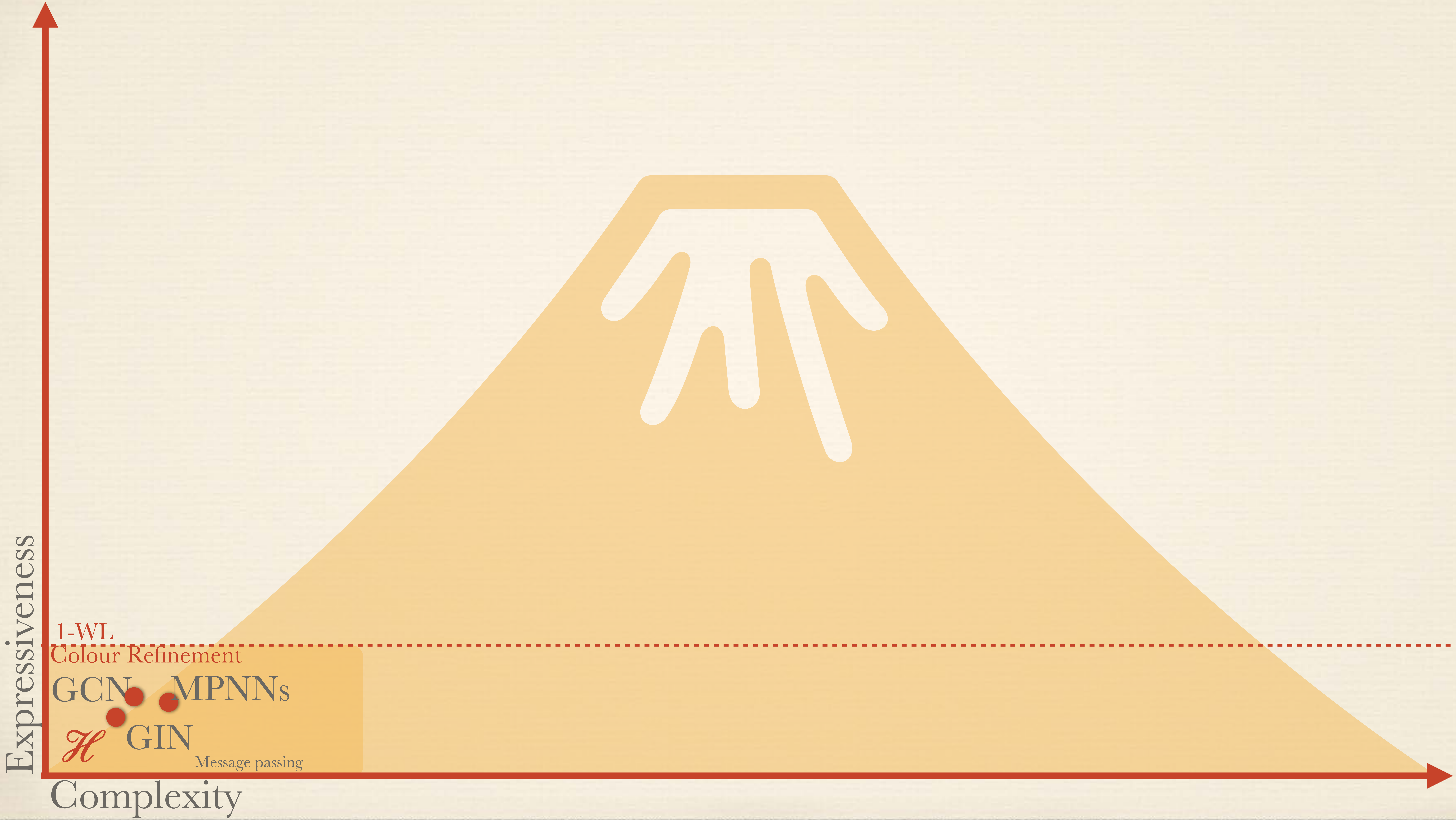
Corollary

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all *trees* T *if and only if* no MPNN can distinguish G from H.

Follows from $\rho(\mathrm{cr}) = \rho(\mathrm{MPNN})$

- MPNNs can only detect tree information from a graph!

Z. Dvorák: *On recognizing graphs by numbers of homomorphisms* (2010)
Dell et al. *Lovász meets Weisfeiler and Leman* (2018)



Expressiveness
1-WL
Colour Refinement
GCN
MPNNs
$\mathscr{H}$
GIN
Message passing
Complexity

Beyond distinguishing power?

- Logical expressiveness
- Approximation properties (universality)

Colour refinement (again)

It was mentioned that ρ(colour refinement) has many characterisations.

Of interest is also a logical one, in particular First-order logic with 2 variables and counting quantifiers ($\boxed{C_2}$).

$$\varphi(x) = \boxed{\exists^{\leq 5} y} \left(E(x, y) \wedge \boxed{\exists^{\geq 2} x} \left(E(y, x) \wedge L_a(x) \right) \right)$$

binary edge predicate   unary label predicate

Given graph G, vertex $v \in V_G$ satisfies φ:

$$\boxed{(G, v) \vDash \varphi}$$

Cai et al.: *An optimal lower bound on the number of variables for graph identifications.* (1992)
M. Grohe. *The logic of graph neural networks* (2021)

Colour refinement (again)

It was mentioned that ρ(colour refinement) has many characterisations.

Of interest is also a logical one, in particular First-order logic with 2 variables and counting quantifiers (C_2).

$$\varphi(x) = \exists^{\leq 5} y \left(E(x,y) \wedge \exists^{\geq 2} x \left(E(y,x) \wedge L_a(x) \right) \right)$$

binary edge predicate　　unary label predicate

Given graph G, vertex $v \in V_G$ satisfies φ: It has at most 5 neighbours

$(G, v) \models \varphi$ each with at least to neighbours labeled "a"

Cai et al.: *An optimal lower bound on the number of variables for graph identifications.* (1992)
M. Grohe. *The logic of graph neural networks* (2021)

Colour refinement and C_2

Theorem (Cai et al. 1992)

Two vertices in a graph have the same colour after t iterations of colour refinement *if and only if* these vertices satisfy the same unary C_2 formulas of quantifier depth t

$$\rho(\text{colour refinement}) = \rho(\text{MPNNs}) = \rho(\mathsf{C}_2)$$

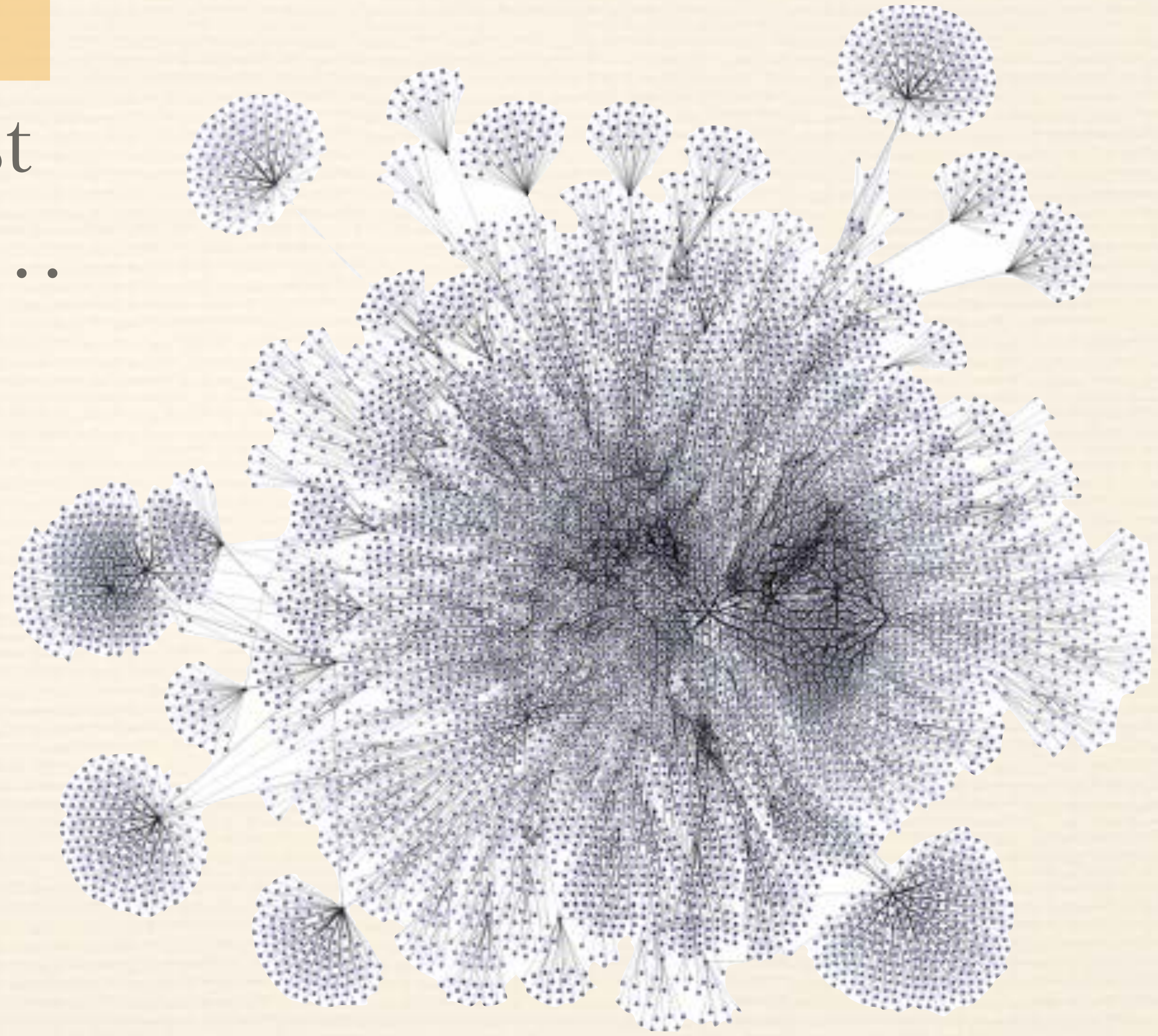
Cai et al.: *An optimal lower bound on the number of variables for graph identifications.* (1992)
M. Grohe. *The logic of graph neural networks* (2021)

Which unary C_2 formulas can MPNNs express?

$\mathcal{H}$ can $\mathcal{C}$-express Ξ if there exists a $\xi \in \mathcal{H}$ such that for all $G \in \mathcal{C}, \mathbf{v} \in V_G^p$:
$\xi(G, \mathbf{v}) = \Xi(G, \mathbf{v})$

- Not all: $\varphi(x) := L_b(x) \wedge \exists y L_r(y)$

I am blue and there exist
a red vertex somewhere…

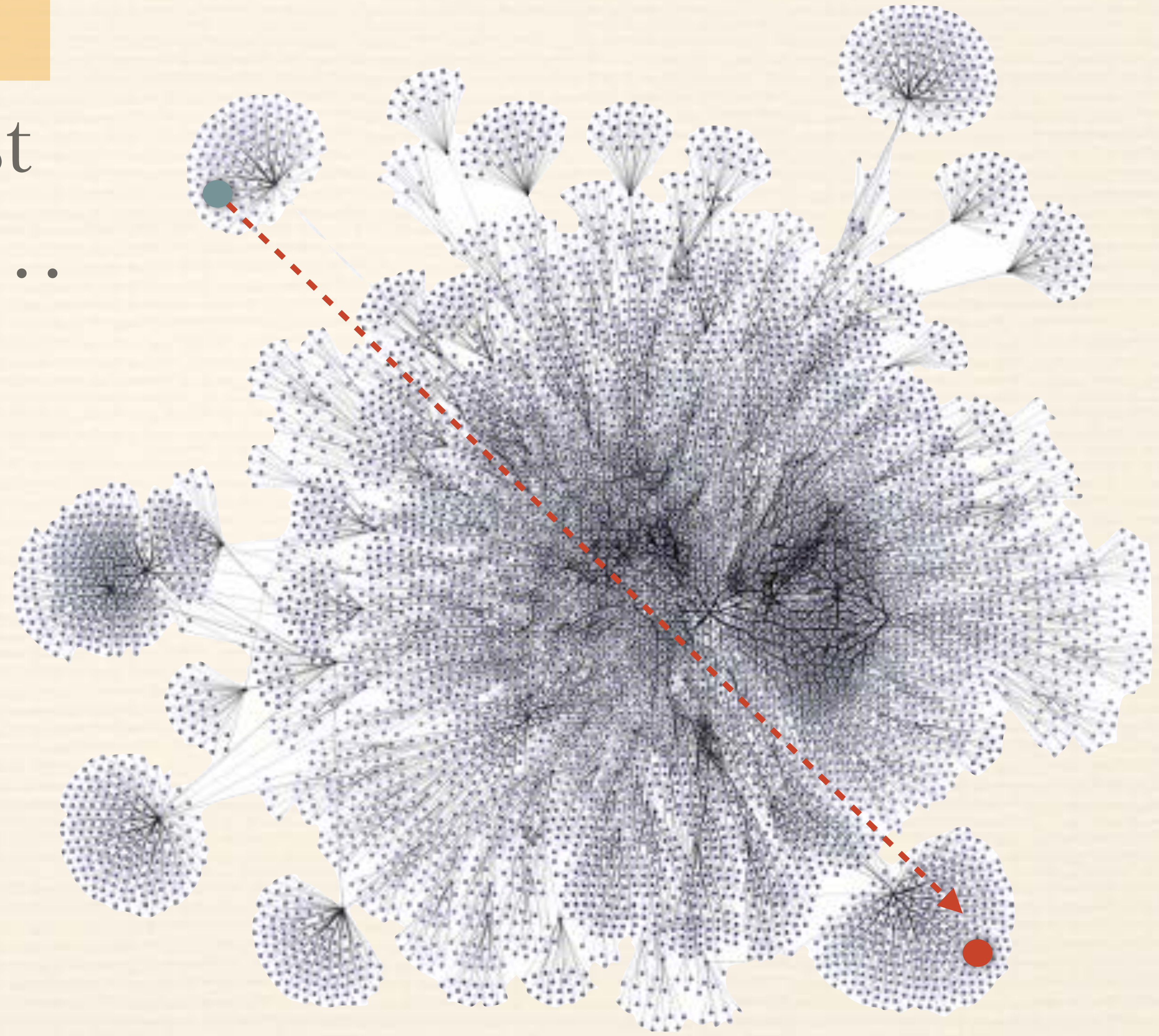


Which unary C_2 formulas can MPNNs express?

$\mathscr{H}$ can $\mathscr{C}$-express Ξ if there exists a $\xi \in \mathscr{H}$ such that for all $G \in \mathscr{C}, \mathbf{v} \in V_G^p$:
$\xi(G, \mathbf{v}) = \Xi(G, \mathbf{v})$

- Not all: $\varphi(x) := L_b(x) \wedge \exists y L_r(y)$

I am blue and there exist a red vertex somewhere…



Cannot be reached by message passing!

Which unary C_2 formulas can MPNNs express?

- Not all: $\varphi(x) := L_b(x) \wedge \exists y L_r(y)$

- Graded modal logic: syntactical fragment of C_2 in which quantifiers are of the form $\exists^{\geq N}\big(E(x,y) \wedge \varphi'(y)\big)$

Theorem (Barceló et al. 2020)

Let $\varphi(x)$ be a unary C_2 formula. Then, $\varphi(x)$ is equivalent to a graded modal logic formula *if and only if* $\varphi(x)$ is expressible by the class of MPNNs.

$$\exists \xi \in \text{MPNNs} : \forall G \in \mathcal{G}, \forall v \in V_G : (G, v) \models \varphi \Leftrightarrow \xi(G, v) = 1$$

Barceló et al.: *The logical expressiveness of graph neural networks* (2020)

MPNN+: Extended MPNNs

- Can we extend MPNNs such that all C_2 formulas (including $\varphi(x) := L_b(x) \wedge \exists y L_r(y)$) can be expressed?

$$\xi^{(t)}(G,v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G,v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G,v), \xi^{(t)}(G,u) \mid u \in N_G(v)\}\}\big)\Big)$$

Add global aggregation in every layer

$$\xi^{(t)}(G,v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G,v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G,v), \xi^{(t)}(G,u) \mid u \in N_G(v)\}\}\big)\, \mathsf{Read}^{(t)}\big(\{\{\xi^{(t)}(G,u) \mid u \in V_G\}\}\big)\Big)$$

Barceló et al.: *The logical expressiveness of graph neural networks* (2020)

MPNNs+

Theorem (Barceló et al. 2020)

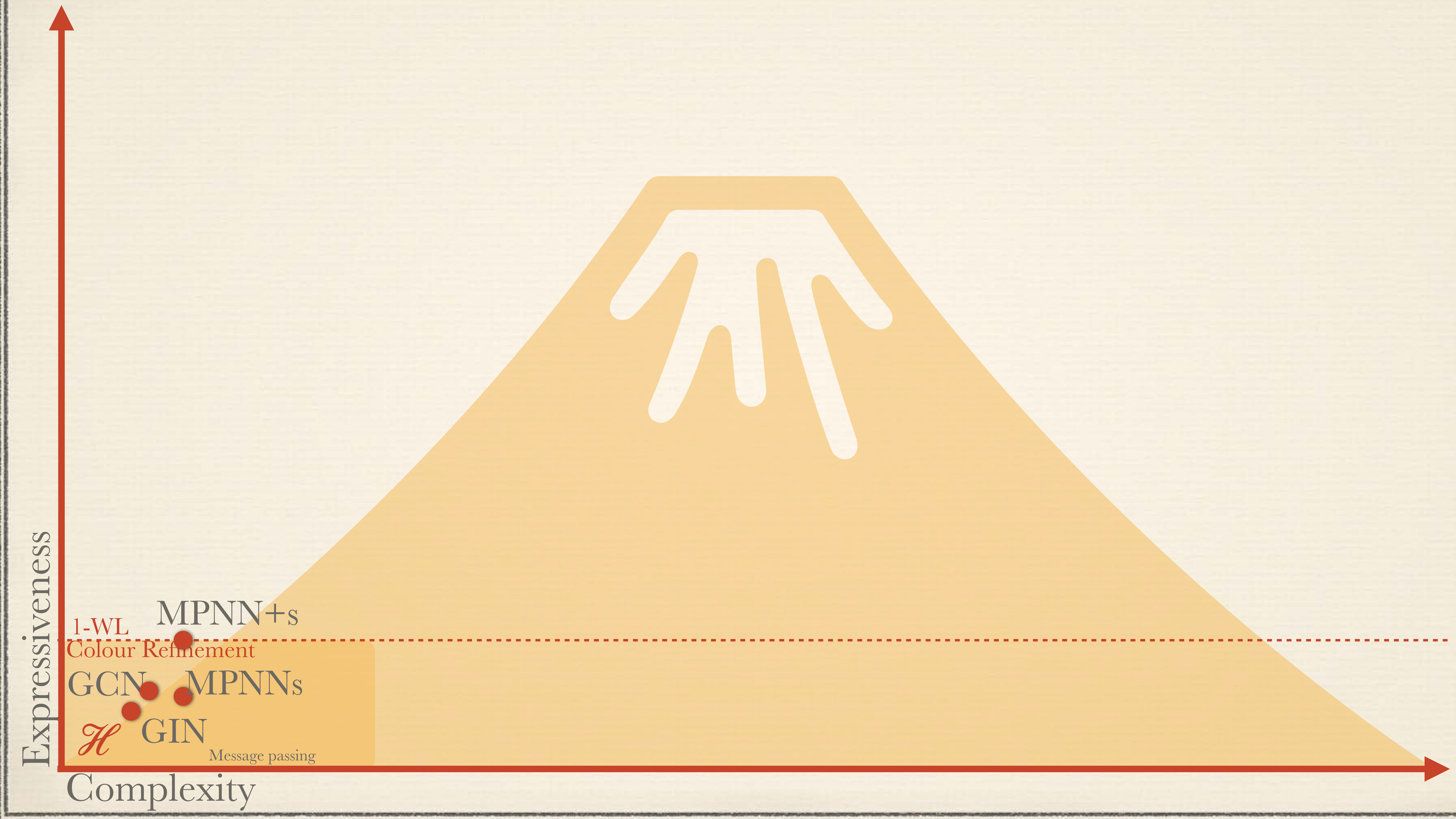
Every unary C_2 formula $\varphi(x)$ is expressible by the class of MPNNs+

- The corresponding colour refinement version is known as the one-dimensional Weisfeiler-Leman algorithm or 1-WL

Can MPNN+ express more formulas? Open problem.

$$\rho(\text{1-WL}) = \rho(\text{MPNNs+})$$

Barceló et al.: *The logical expressiveness of graph neural networks* (2020)



MPNN+s
1-WL
Colour Refinement
GCN
MPNNs
GIN
$\mathscr{H}$
Message passing
Expressiveness
Complexity

Approximation properties

- Equip set of graphs $\mathcal{G}$ with a topology and assume that $\mathcal{H}$ consist of continuous graph embeddings from $\mathcal{G}$ to $\mathbb{R}$.

- Let $\mathcal{C} \subseteq \mathcal{G}$ be a compact set of graphs.

Stone-Weierstrass

Theorem (Azizian & Lelarge 2021, G. and Reutter 2022)

If $\mathcal{H}$ is closed under linear combinations and product, then $\mathcal{H}$ can $\mathcal{C}-$approximate any continuous function $\Xi : \mathcal{C} \to \mathbb{R}$ satisfying $\rho(\mathcal{H}) \subseteq \rho(\{\Xi\})$.

- Can be generalised to general embeddings with output space $\mathbb{R}^d$

W. Azizian and M. Lelarge: *Characterizing the expressive power of invariant and equivariant graph neural networks* (2021)
G. and J. Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)

MPNNs: Approximation

Theorem (Azizian & Lelarge 2021, G. and Reutter 2022)

On compact set of graphs, MPNNs can approximate any continuous graph embedding $\Xi : \mathscr{C} \to \mathbb{R}$ satisfying $\rho(\text{colour refinement}) \subseteq \rho(\{\Theta\})$

- We know $\rho(\text{MPNNs}) = \rho(\text{colour refinement})$
- Update functions can be used to approximate product and take linear combinations of MPNNs

- Intricate relation between distinguishing power and approximation properties

W. Azizian and M. Lelarge: *Characterizing the expressive power of invariant and equivariant graph neural networks* (2021)
G. and J. Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)

Universality and graph isomorphism

Theorem (Chen et al. (2019)

In order for a class of methods to be able o approximate any (invariant) continuous functions, the class of methods should be able to distinguish any two non-isomorphic graphs.

Proof

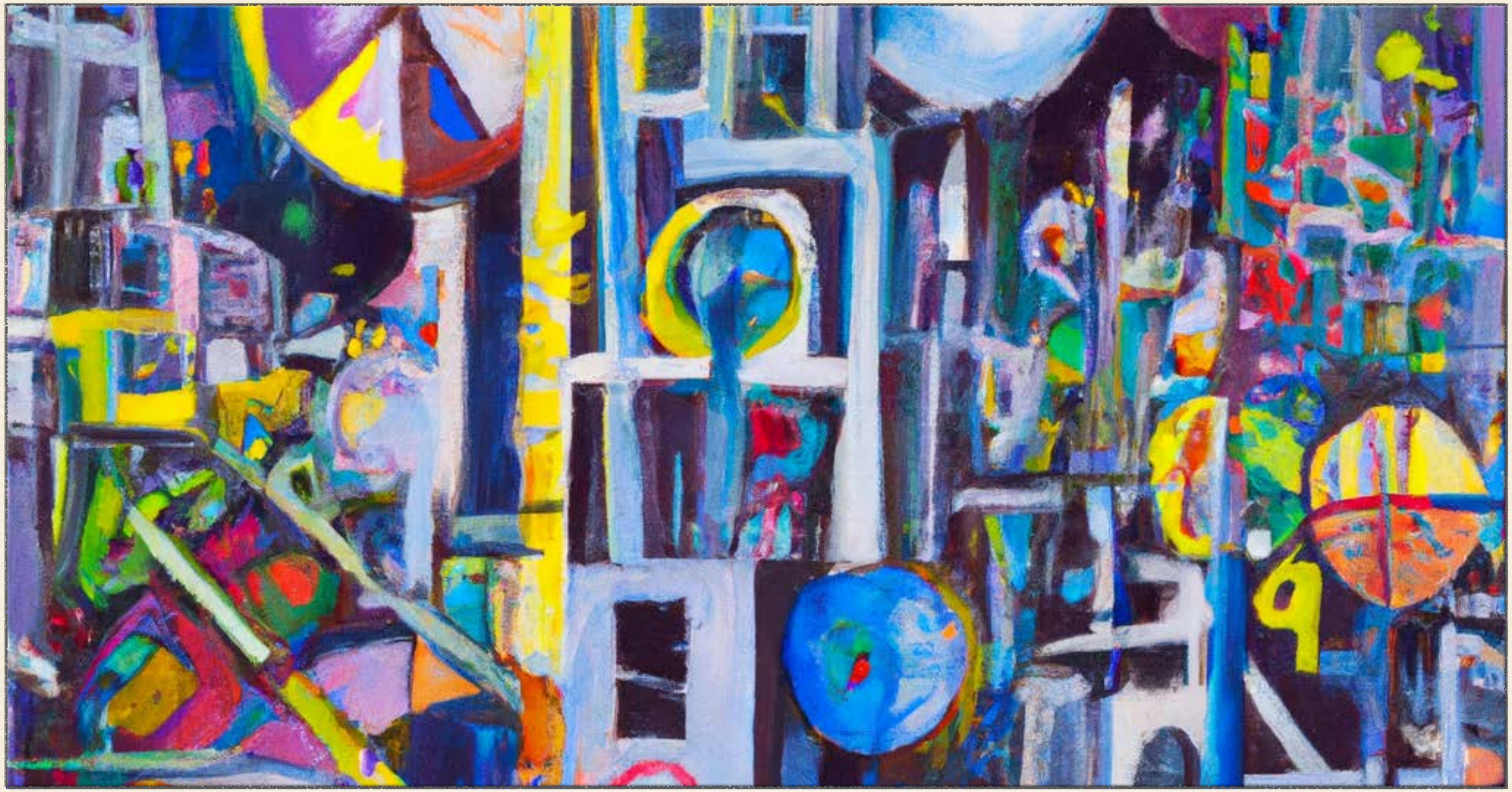
Minimal size $\rho(\mathscr{H}) \subseteq \rho(\{\Xi\})$

$$\uparrow$$

$$(G, H) \in \rho(\mathscr{H}) \Leftrightarrow G \cong H$$

Chen et al.: *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

Questions?



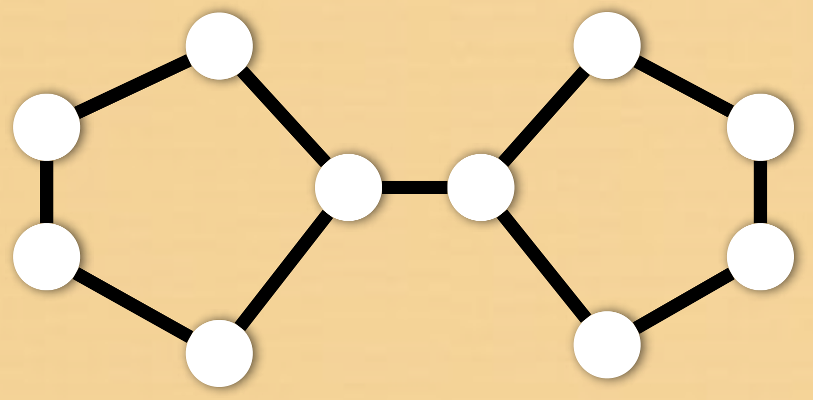
Feature Augmentation

Boost the expressive power by adding information

More expressive MPNNs?

Feature engineering

- Deep learning and MPNNs have replaced “old school” feature engineering approach.



- Number of edges
- Number of cycles of length 5 $\rightarrow \mathbb{R}^d \rightarrow$ SVM
- Centrality measures

- MPNNs were supposed to learn such features automatically …

Idea #1: Adding expressive features

Recall:

Theorem

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all *trees* T if and only if no MPNN can distinguish G from H.

Idea #1: Adding expressive features

Recall:

Theorem

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all *trees* T if and only if no MPNN can distinguish G from H.

- What if we add subgraph information before doing message-passing?

More than trees

Idea #1: Adding expressive features

Recall:

Theorem

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all *trees* T if and only if no MPNN can distinguish G from H.

- What if we add subgraph information before doing message-passing?

More than trees

We will try this out later in the hands-on session!

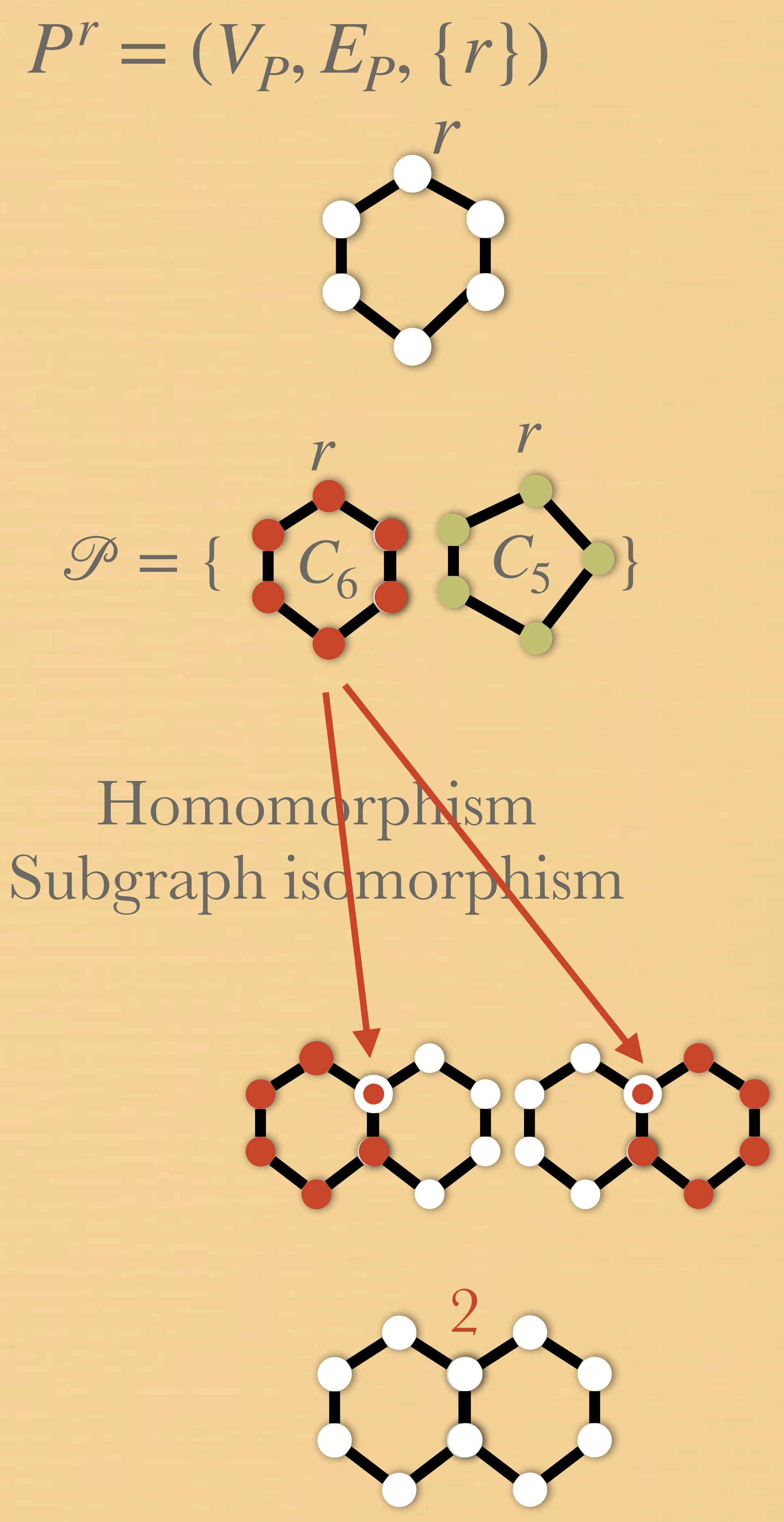
Structural encodings

1. Choose collection of rooted graph patterns/motifs

$$\mathscr{P} := \{P^r_1, \ldots, P^r_\ell\}$$

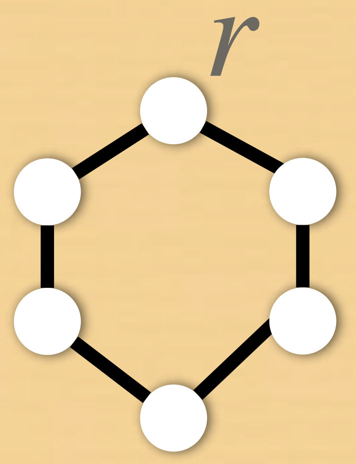
2. Choose how to match subgraphs in $\mathscr{P}$ with data graph G

3. Add count of matches to vertices as extended features.

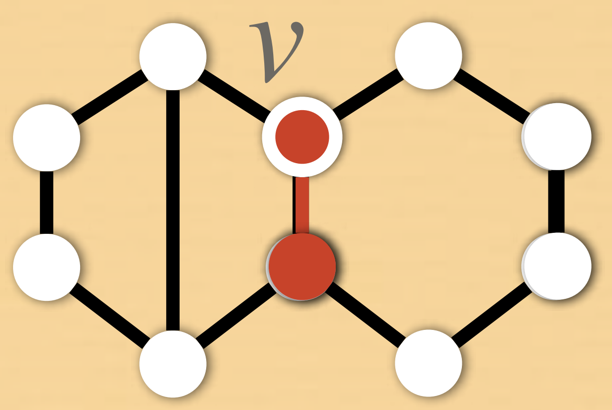


Matches

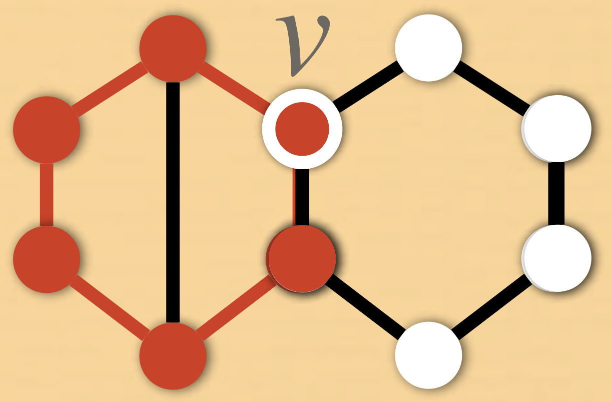
- **Homomorphism**: edge preserving
- **Subgraph isomorphism**: bijection , edge preserving
- **Induced subgraph isomorphism**: bijection, edge preserving (both ways)

$P^r = (V_P, E_P, \{r\})$ 

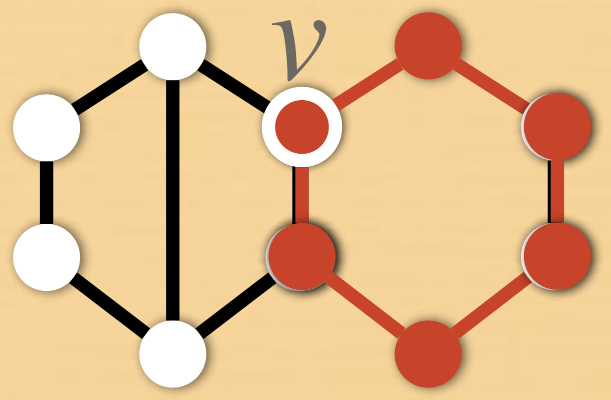
$\pi : V_P \rightarrow V_S \subseteq V_G$ containing v



hom(P^r, G^v)



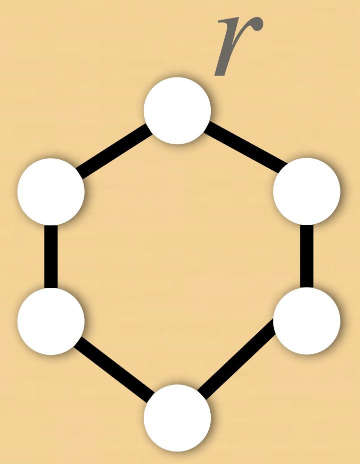
subiso(P^r, G^v)



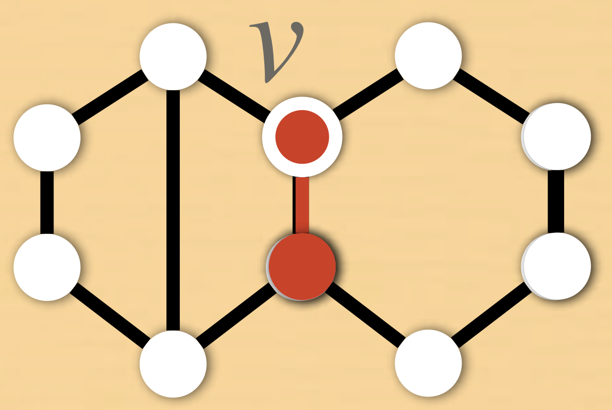
indsubiso(P^r, G^v)

Matches

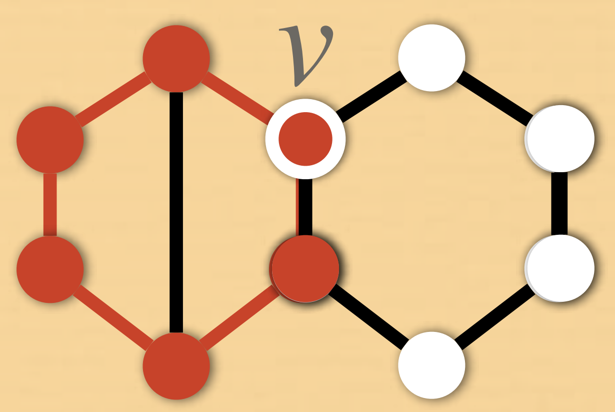
- Homomorphism: edge preserving
- Subgraph isomorphism: bijection , edge preserving
- Induced subgraph isomorphism: bijection, edge preserving (both ways)

$P^r = (V_P, E_P, \{r\})$ 

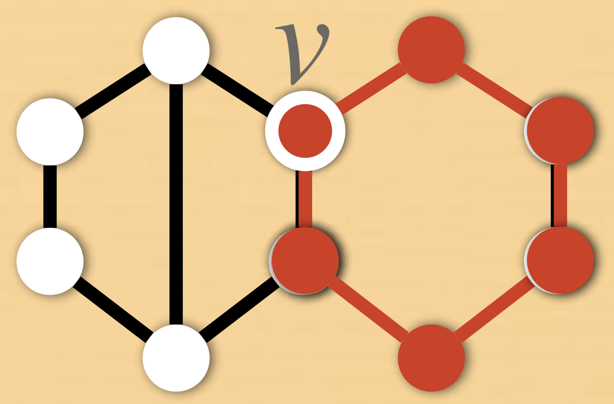
$\pi : V_P \to V_S \subseteq V_G$ containing v



$\mathrm{hom}(P^r, G^v)$



$\mathrm{subiso}(P^r, G^v)$



$\mathrm{indsubiso}(P^r, G^v)$

Counts

$\mathscr{P}$-MPNNs

- Add structural encoding as vertex features and run MPNN

$$\mathscr{P} := \{P_1^r, \ldots, P_\ell^r\}$$

$\mathscr{P}$-MPNNs

$$\xi^{(0)}(G, v) := \text{Hot-one encoding of label of vertex } v + \mathsf{hom}(P_1^r, G^v), \ldots, \mathsf{hom}(P_\ell^r, G^v)$$

$$\xi^{(t)}(G, v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G, v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G, v), \xi^{(t)}(G, u), \mathsf{hom}(P_1^r, G^u), \ldots, \mathsf{hom}(P_\ell^r, G^u) \mid u \in N_G(v)\}\}\big)\Big)$$

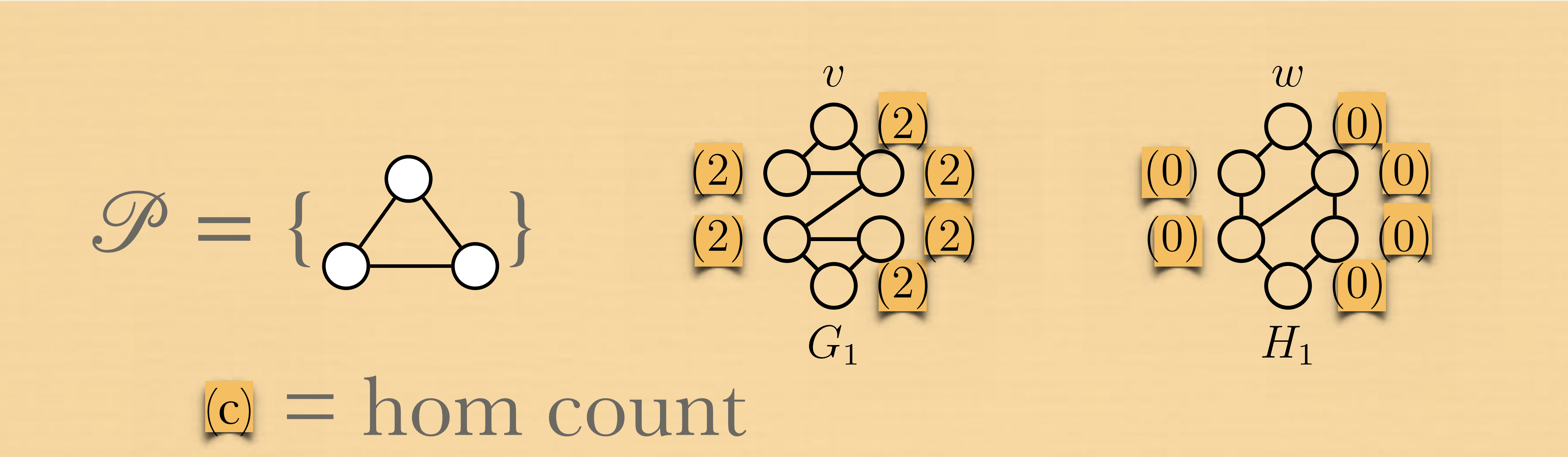
$$\rho(G) := \mathsf{Readout}\Big(\{\{\xi^{(L)}(G, v) \mid v \in V_G\}\}\Big)$$

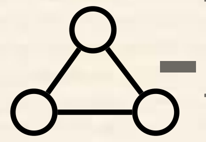
hom counts of patterns

- Did we increase expressive power?

Barceló et al.: *Graph neural networks with local graph parameters.* (2021)
Barceló et al.: *Graph neural networks with local graph parameters.* (2021)

$\mathscr{P}$-MPNNs



- We have seen that these graphs equivalent for colour refinement but clearly not for -MPNNs.

- So, increase in power!

- What is their precise expressive power?

$\mathscr{P}$-MPNNs: Expressive power

Theorem

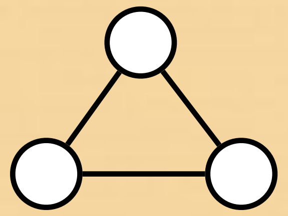
$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all $\mathscr{P}$−*pattern trees* T *if and only if* no P-MPNN can distinguish G from H.

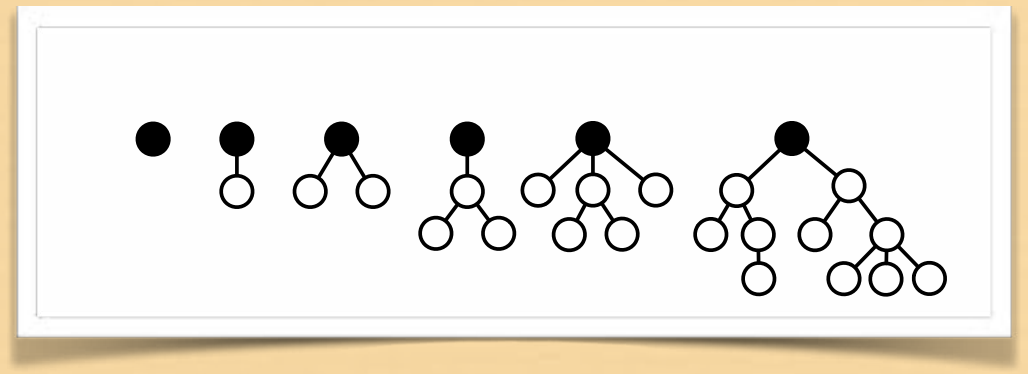
Barceló et al.: *Graph neural networks with local graph parameters.* (2021)

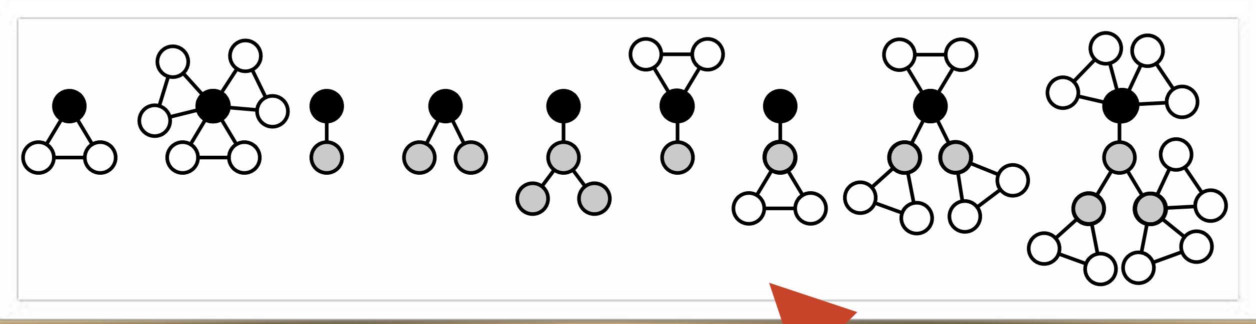
$\mathscr{P}$-MPNNs: Expressive power

Theorem

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all $\mathscr{P}$*–pattern trees* T *if and only if* no P-MPNN can distinguish G from H.

$\mathscr{P} = \{$  $\}$





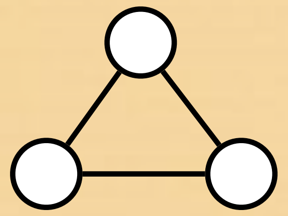
Take tree: add in each tree vertex copies of rooted patterns

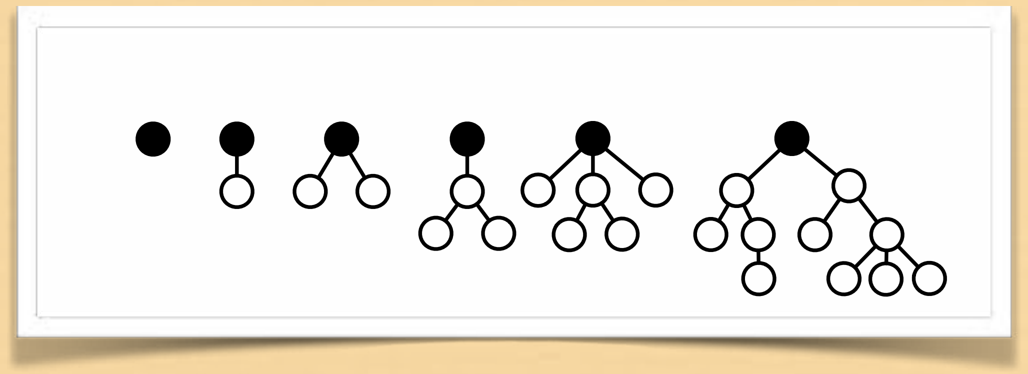
Barceló et al.: *Graph neural networks with local graph parameters.* (2021)

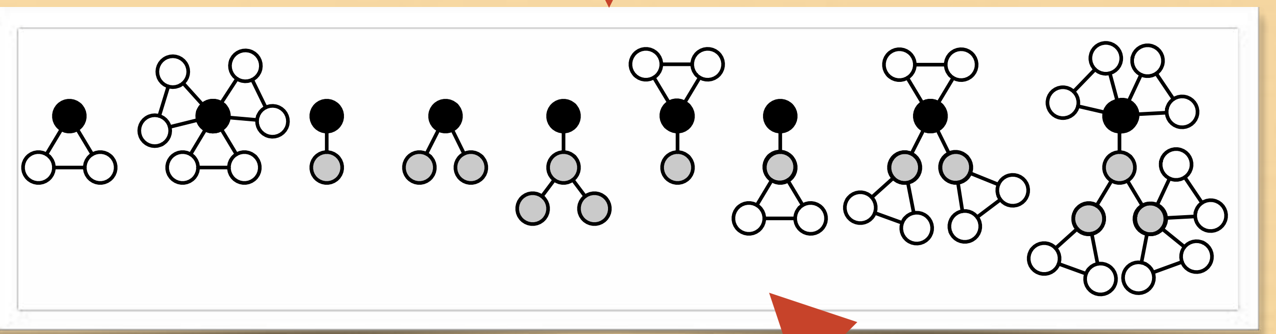
$\mathscr{P}$-MPNNs: Expressive power

Theorem

hom(T, G) = hom(T, H) for all $\mathscr{P}$*−pattern trees T if and only if* no P-MPNN can distinguish G from H.

$\mathscr{P} = \{$  $\}$





Take tree: add in each tree vertex copies of rooted patterns

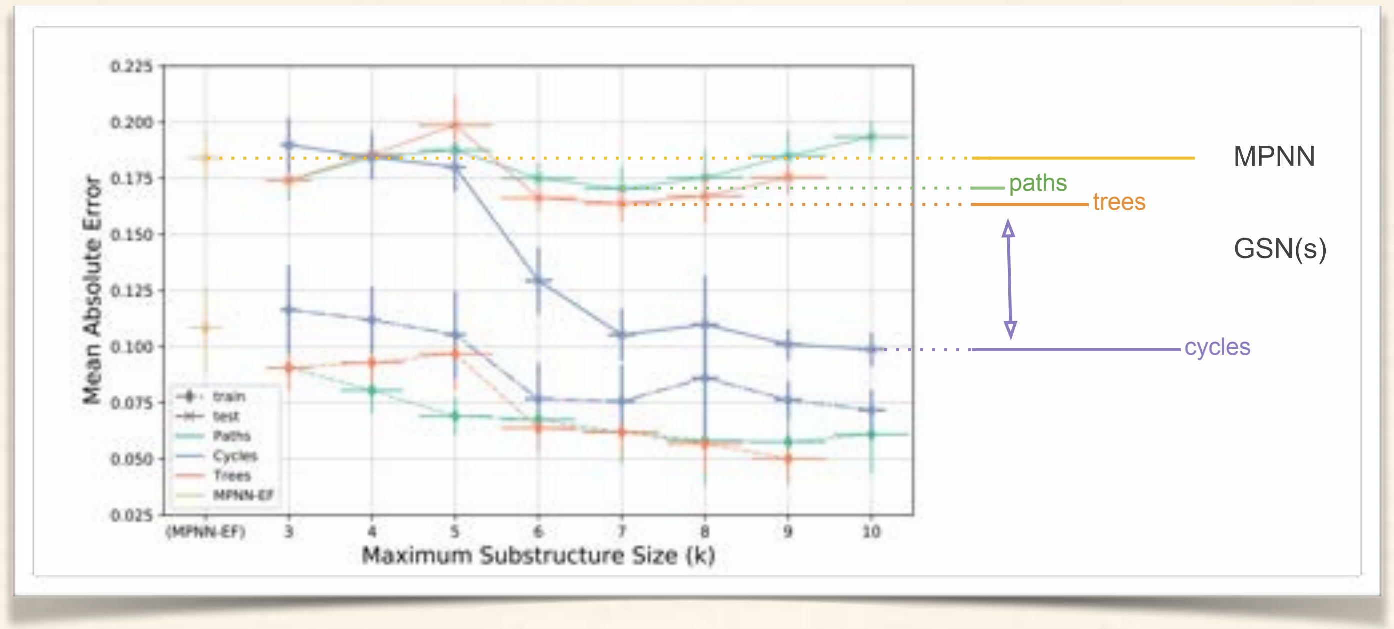
Zinc dataset

Set ($\mathcal{F}$)	MAE
None	0.47±0.02
$\{C_3\}$	0.45±0.01
$\{C_4\}$	0.34±0.02
$\{C_6\}$	0.31±0.01
$\{C_5, C_6\}$	0.28±0.01
$\{C_3, \ldots, C_6\}$	0.23±0.01
$\{C_3, \ldots, C_{10}\}$	**0.22±0.01**

Barceló et al.: *Graph neural networks with local graph parameters.* (2021)

Choice of matching?

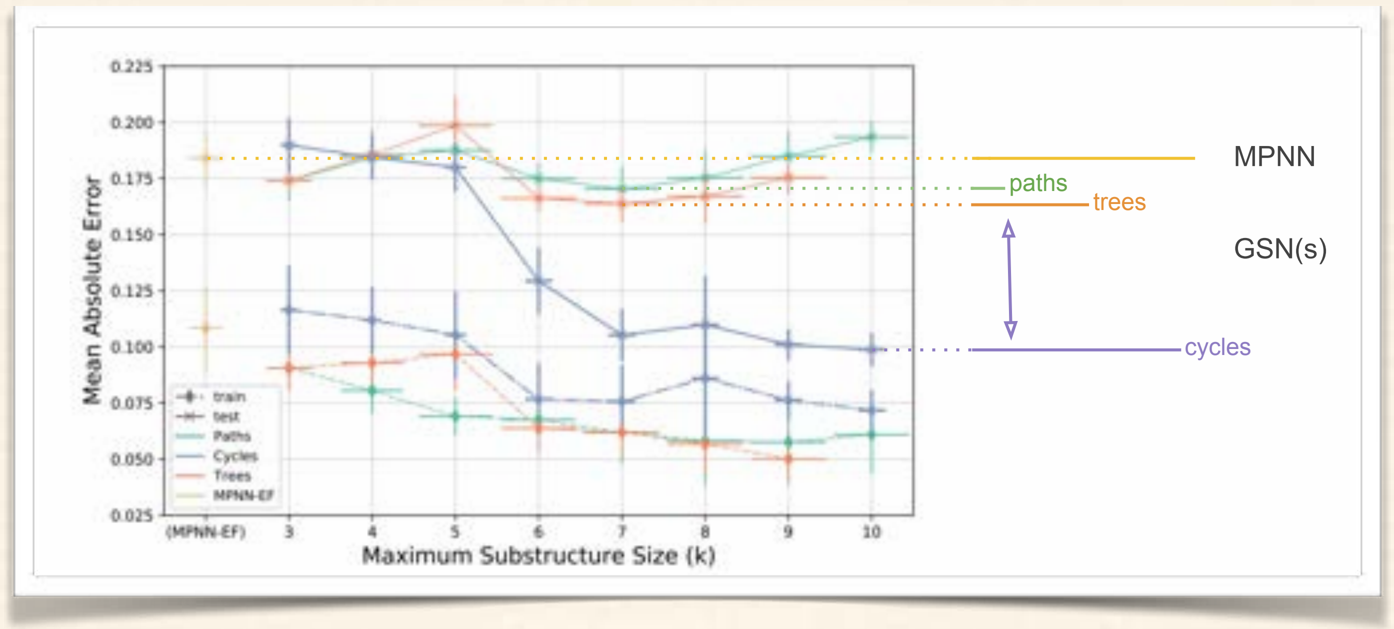
- Graph Substructure Networks (GSNs): use subiso counts.



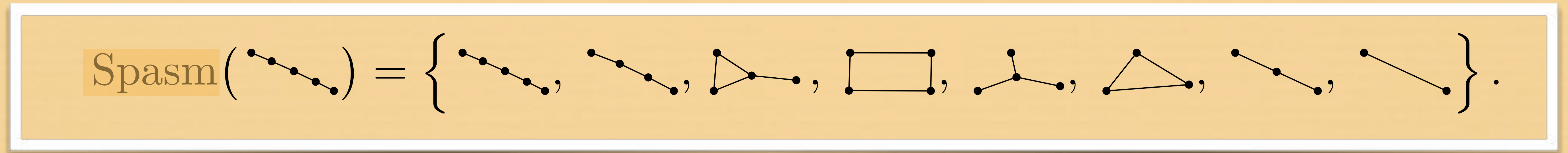
Bouritsas et al.: *Improving graph neural network expressivity via subgraph isomorphism counting* (2020)
Curticapean et al.: *Homomorphisms are a good basis for counting small subgraphs.* (2017)

Choice of matching?

- Graph Substructure Networks (GSNs): use subiso counts.



- Expressive power of GSN? Reduction to homomorphism counts



Curticapean et al. (2017)

$\mathscr{P}$ $\mathscr{P}^\star$

subiso count → hom count of *spasm*

Similar connection between for other matchings

More hom counts needed for same subgraph iso :-(

Bouritsas et al.: *Improving graph neural network expressivity via subgraph isomorphism counting* (2020)
Curticapean et al.: *Homomorphisms are a good basis for counting small subgraphs.* (2017)

GSNs: Expressive power

Theorem

If $\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all $\mathscr{P}^{\star}-$*pattern trees* T, then no SGN can distinguish G from H.

- A direct characterisation in terms of subiso is also possible.
- The choice of patterns in $\mathscr{P}$ is crucial
- Simple patterns such as cycles and cliques work well.

The larger and complex $\mathscr{P}$ $\Rightarrow$ more complexity counting

$\Rightarrow$ more expressive power


The larger and complex 𝒫 ⇒ more complexity counting
⇒ more expressive power
We
will come back
to this later
𝒫⋆-MPNNs
𝒫-SGNs
𝒫-MPNNs
MPNN+s
1-WL
Colour Refinement
GCN
MPNNs
ℋ
GIN
Message passing
Expressiveness
Complexity

Idea #2: (Random) Vertex identifiers

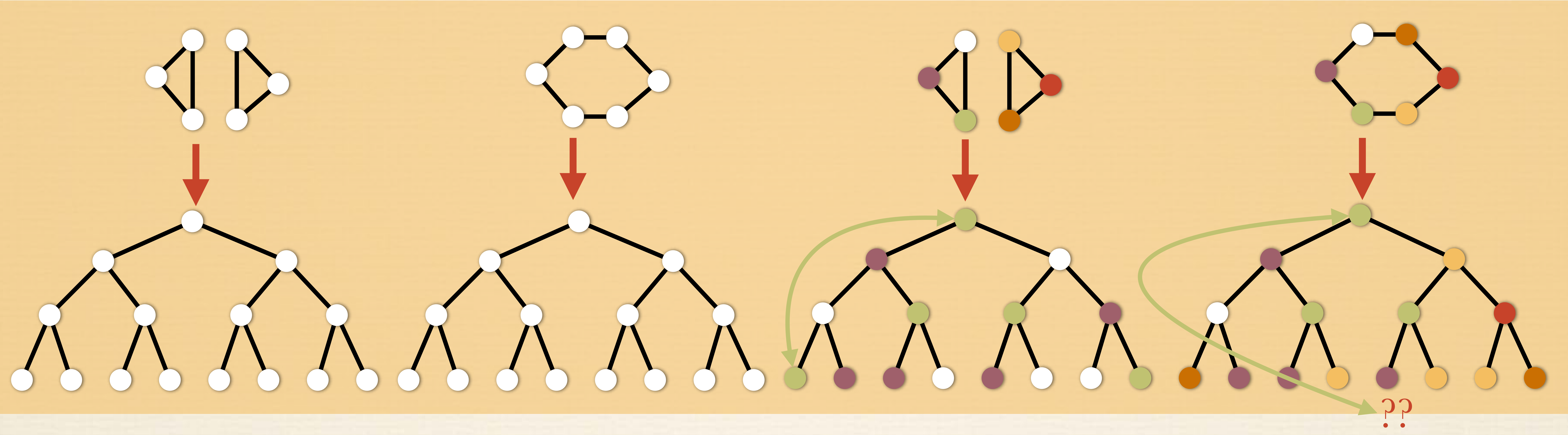
- Message-Passing is only based on vertex features and adjacency information.

- Two different vertices with the same vertex features will be treated the same (if they have the same colour in colour refinement).

What if we add vertex identifiers?

Vertex identifiers

Self identification: useful for cycle detection

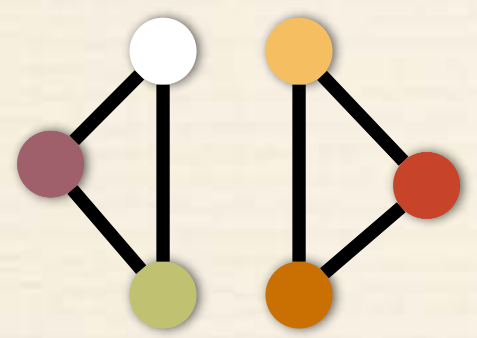


In terms of colour refinement: every vertex has a unique colour

Logic comes to rescue

Theorem (Cai et al. 1992)

Recall: Two vertices in a graph have the same colour after t iterations of colour refinement *if and only if* these vertices satisfy the same unary C_2 formulas of



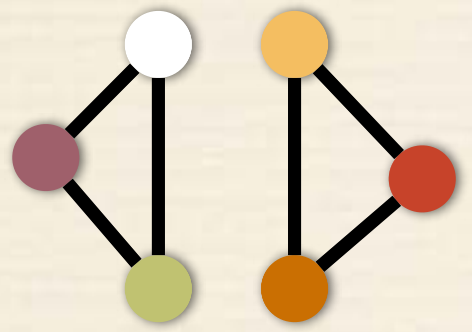
If every vertex has a unique colour, then can be identified with a C_2 formula

We can express in C_2 a formula φ_G satisfying

$$H \models \varphi_G \iff H \cong G$$

Cai et al.: *An optimal lower bound on the number of variables for graph identifications.* (1992)

Logic comes to rescue



$$\alpha_v(x) := \bigwedge_{c \text{ id of } v} \mathsf{Lab}_c(x) \wedge \bigwedge_{c' \text{ is not id of } v} \neg\mathsf{Lab}_{c'}(x)$$

$$\beta_{v,w}(x,y) := \begin{cases} \alpha_v(x) \wedge \alpha_w(y) \wedge E(x,y) & (v,w) \in E_G \\ \alpha_v(x) \wedge \alpha_w(y) \wedge \neg E(x,y) & (v,w) \notin \in E_G \end{cases}$$

$$\varphi_G := \bigwedge_{v \in V_G} (\exists x \alpha_v(x) \wedge \neg \exists^{\geq 2} x \alpha_v(x)) \wedge \bigwedge_{v,w \in V_G} \exists x \exists y \beta_{v,w}(x,y)$$

$$\boxed{H \models \varphi_G \iff H \cong G}$$

MPNNs+ and vertex ids

Recall: Theorem

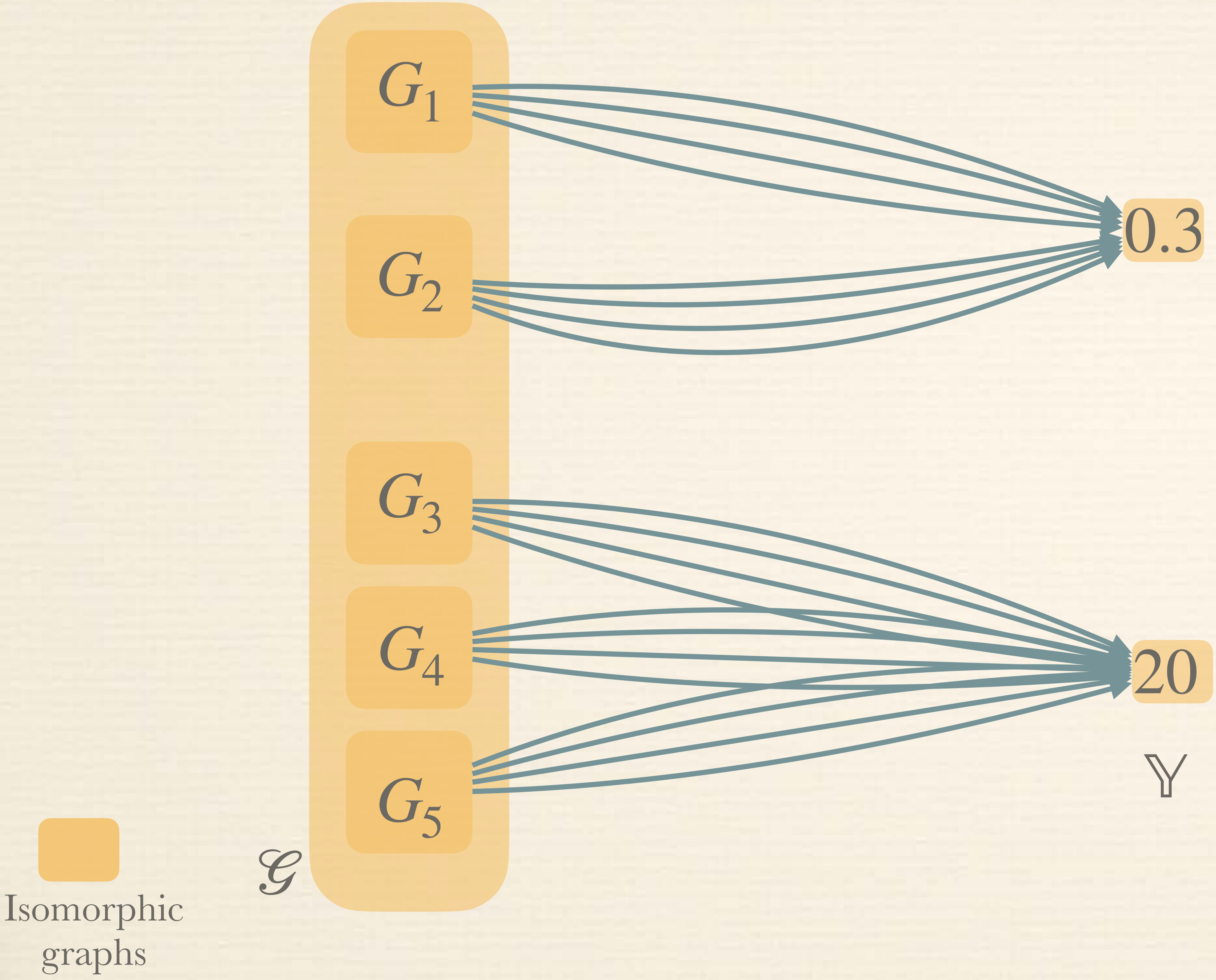
Every C_2 formula is expressible by the class of MPNNs+

Idea: We use MPNNs+ to express φ_G

$$\varphi_G := \bigwedge_{v \in V_G} (\exists x \alpha_v(x) \wedge \neg \exists^{\geq 2} x \alpha_v(x)) \wedge \bigwedge_{v,w \in V_G} \exists x \exists y \beta_{v,w}(x, y)$$

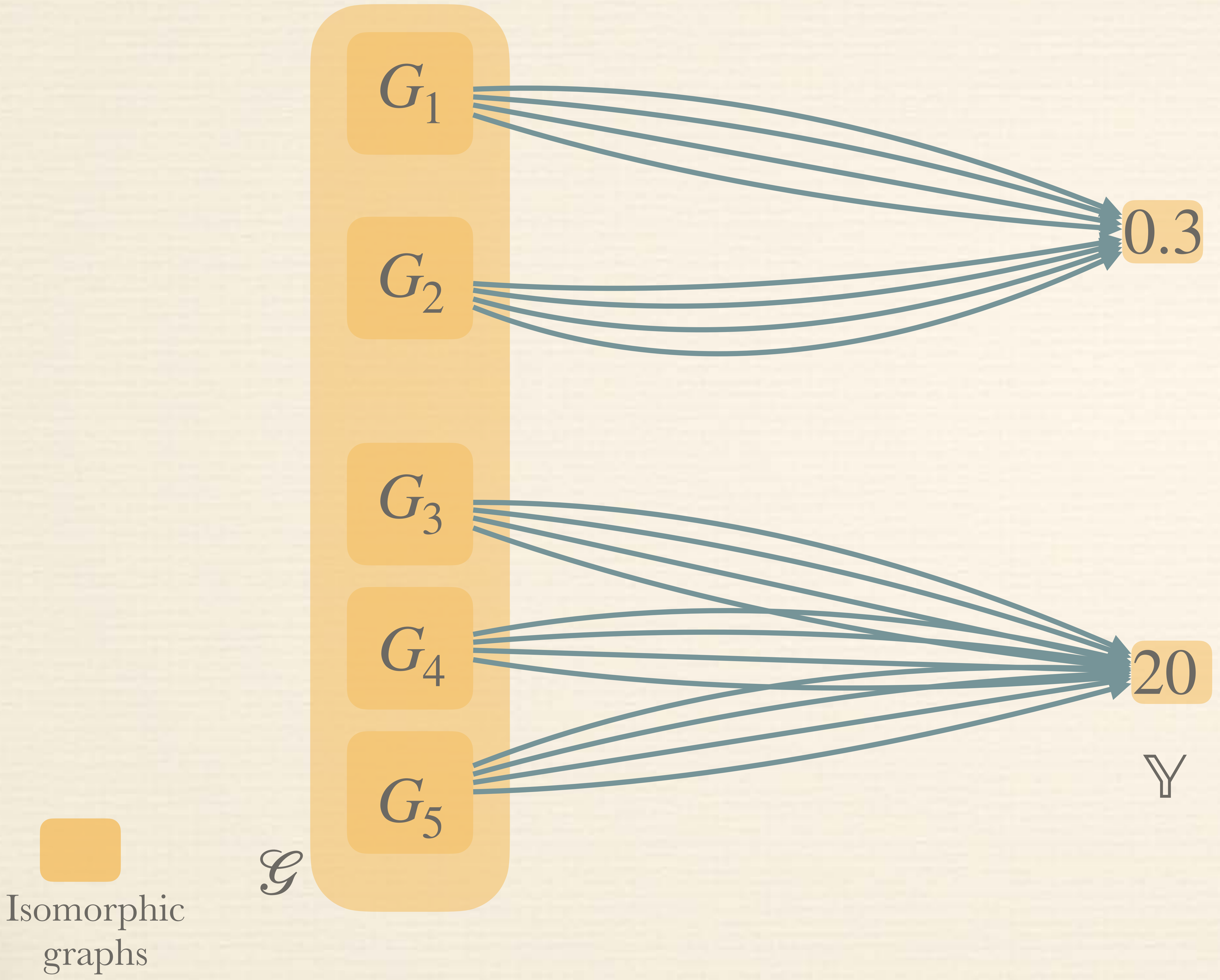
Barceló et al.: *The logical expressiveness of graph neural networks* (2020)
Abboud et al. :*The surprising power of graph neural networks with random node initialization.* (2021)

MPNNs+ and vertex ids



Abboud et al. : *The surprising power of graph neural networks with random node initialization.* (2021)

MPNNs+ and vertex ids

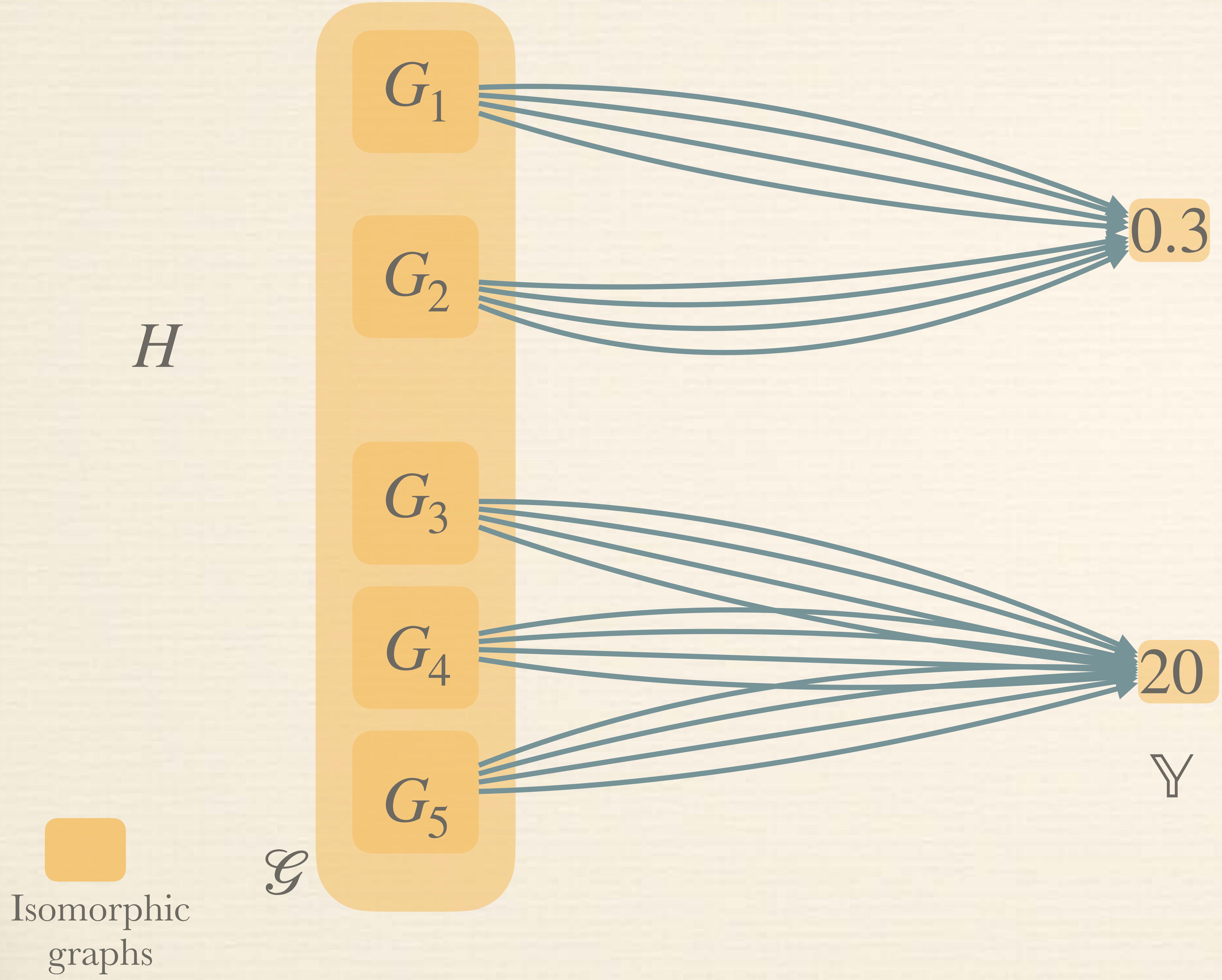


$$\psi := \Big(\bigvee_{i=1,2} \varphi_{G_i}\Big) \times 0.3 + \Big(\bigvee_{i=3,4,5} \varphi_{G_i} \times 20$$

MPNN+

Abboud et al. :*The surprising power of graph neural networks with random node initialization.* (2021)

MPNNs+ and vertex ids



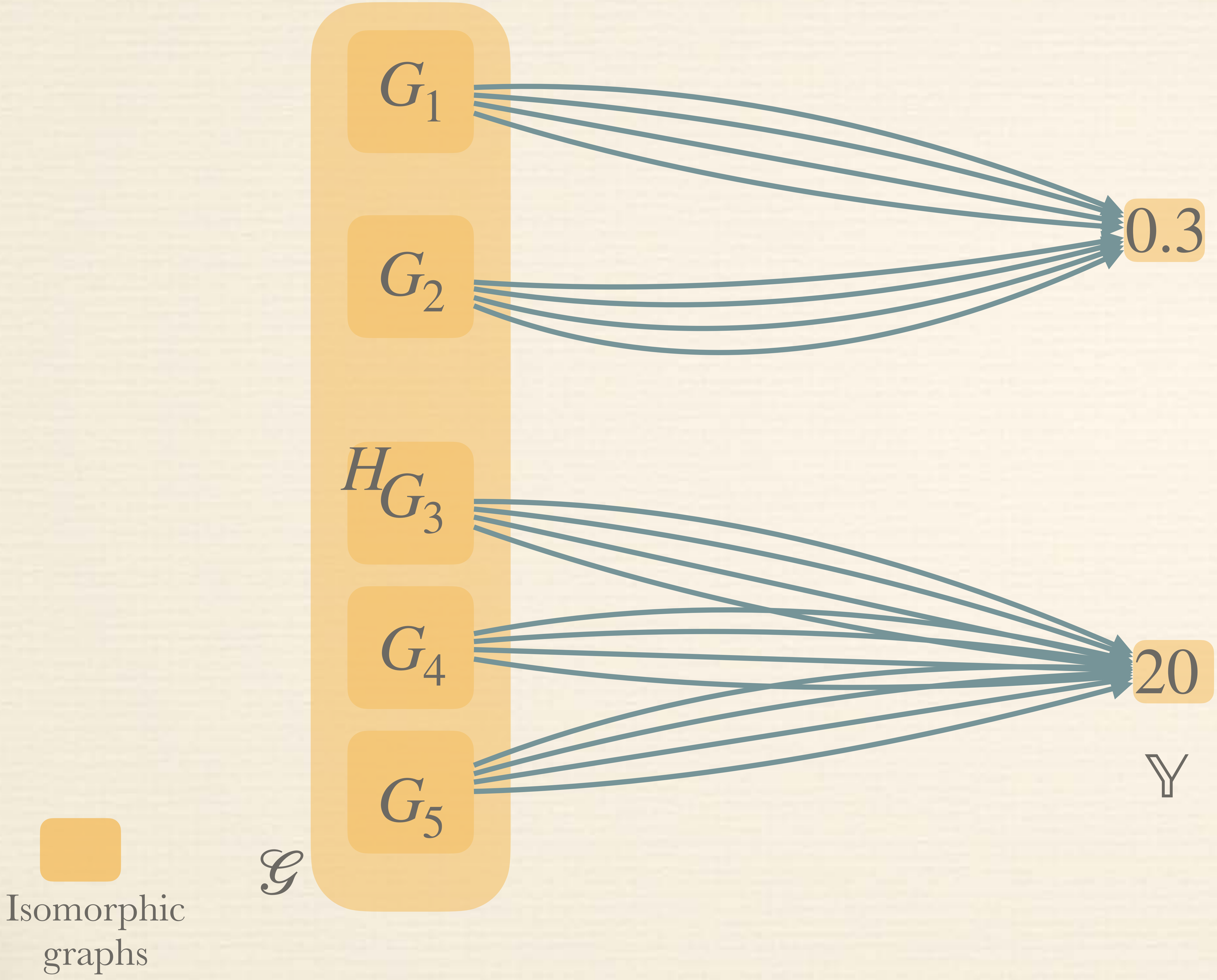
$$\psi := (\bigvee_{i=1,2} \varphi_{G_i}) \times 0.3 + (\bigvee_{i=3,4,5} \varphi_{G_i} \times 20$$

MPNN+

Given graph H

Abboud et al. :*The surprising power of graph neural networks with random node initialization.* (2021)

MPNNs+ and vertex ids



$$\psi := \Big(\bigvee_{i=1,2} \varphi_{G_i}\Big) \times 0.3 + \Big(\bigvee_{i=3,4,5} \varphi_{G_i} \times 20$$

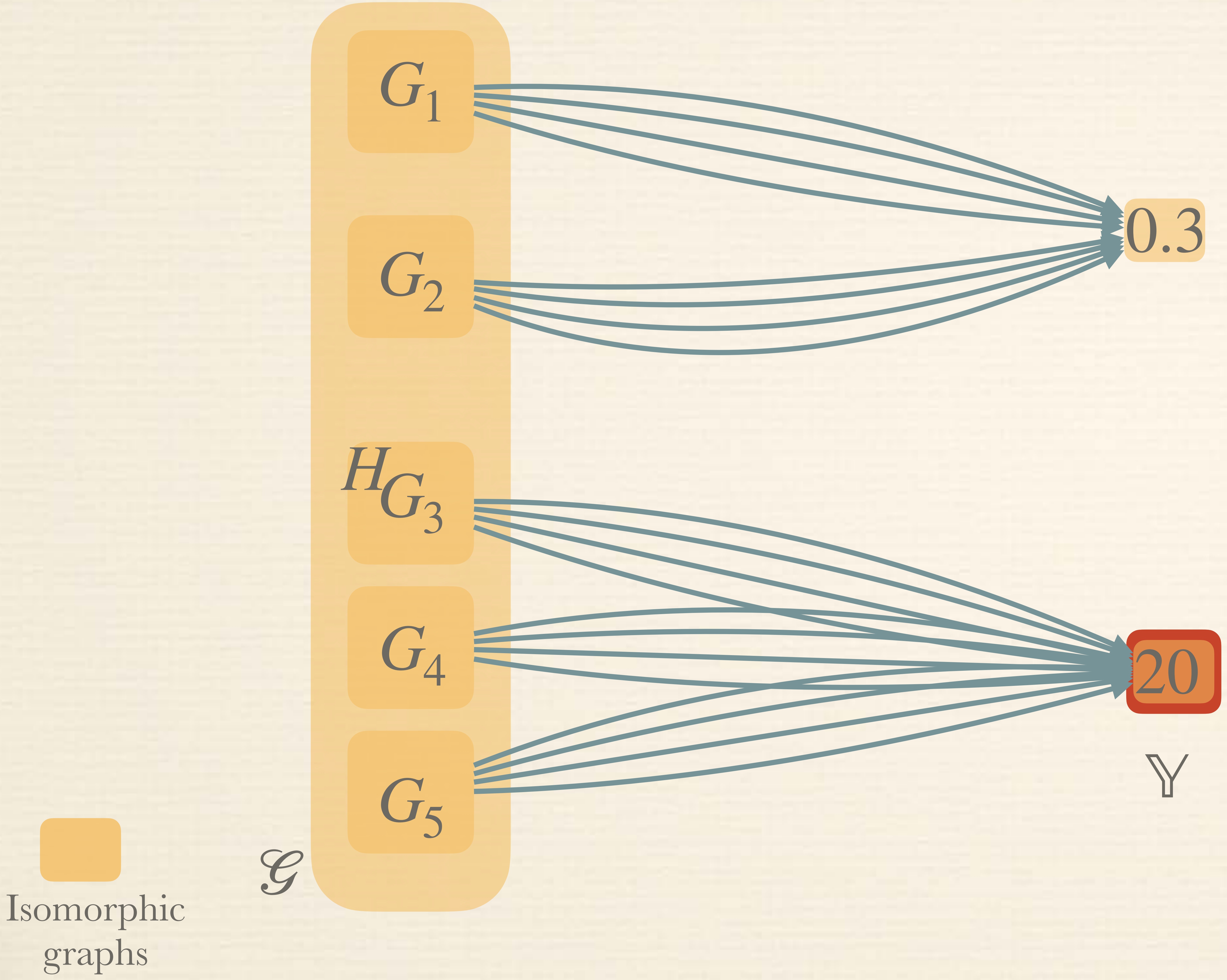
MPNN+

Given graph H

MPNN+ selects i s.t. $H \models \varphi_{G_i}$

Abboud et al. : *The surprising power of graph neural networks with random node initialization.* (2021)

MPNNs+ and vertex ids



$$\psi := \Big(\bigvee_{i=1,2} \varphi_{G_i}\Big) \times 0.3 + \Big(\bigvee_{i=3,4,5} \varphi_{G_i} \times 20$$

MPNN+

Given graph H

MPNN+ selects i s.t. $H \models \varphi_{G_i}$

MPNN+ looks up value for G_i

Abboud et al. : *The surprising power of graph neural networks with random node initialization.* (2021)

rMPNNs+

- How to choose identifiers? Common choice is at random!
- With high probability random features are vertex identifiers

Theorem

rMPNNs(+) approximate any invariant graph/ vertex embedding with high probability

- Invariance of computed embedding only in expectation!

Dasoulas et al.: *Coloring graph neural networks for node disambiguation* (2020)
Abboud et al. :*The surprising power of graph neural networks with random node initialization.* (2021)
Sato et al.: *Random features strengthen graph neural networks* (2021).

Invariance by averaging

- Add vertex identifiers $G \mapsto (G, \mathsf{id})$
- Take embedding method $\chi \in \mathcal{H}$
- All permutation $\pi \in S_n$ with $n = |V_G|$
- Average/Aggregate $P = S_n$:

$$\xi(G) := \frac{1}{|P|} \sum_{\pi \in P} \xi(\pi(G, \mathsf{id}))$$

$$\xi(G) := \max_{\pi \in P} \xi(\pi(G, \mathsf{id}))$$

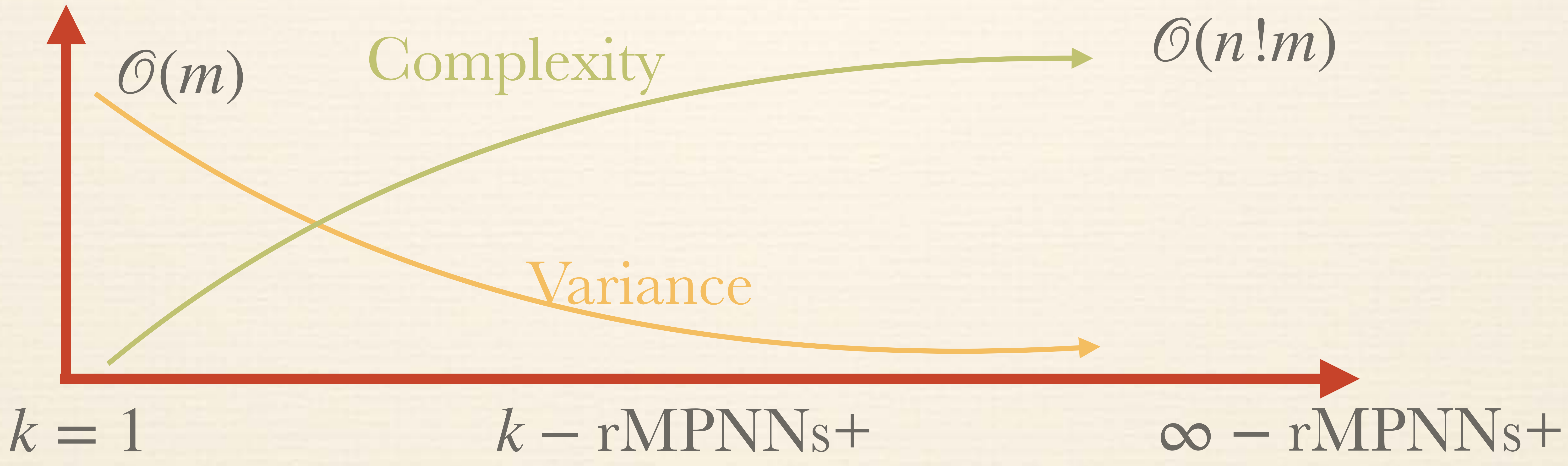
Dasoulas et al.: *Coloring graph neural networks for node disambiguation* (2020)
Murphy et al.: *Relational pooling for graph representations.* (2019)

Partial averaging, k-rMPNNs+

Loose interpretation of k-CLIP

- Let $P \subseteq S_n$ of size $|P| = k$

$$\xi(G) := \max_{\pi \in P} \xi(\pi(G, \text{id}))$$



Dasoulas et al.: *Coloring graph neural networks for node disambiguation* (2020)

1-rMPNN+
randomness clouds
k-rMPNN+
∞ − rMPNN+
𝒫⋆-MPNNs
𝒫-SGNs
𝒫-MPNNs
MPNN+s
1-WL
Colour Refinement
GCN
MPNNs
ℋ
GIN
Message passing
Expressiveness
Complexity
not invariant

Idea #3: Use global information

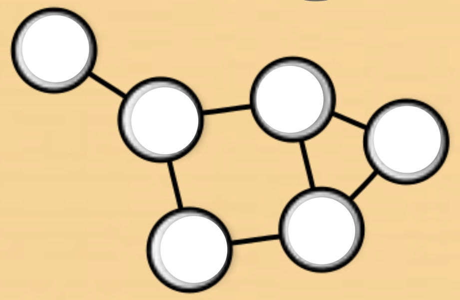
- Extract global graph information and use it as positional encodings of vertices
  - Spectral information
  - Shortest paths (distance information)
  - Biconnectivity (connectivity information)

Kreuzer et al.: *Rethinking graph transformers by spectral attention* (2021)
Ying et al.: *Do transformers really perform bad for graph representation* (2021)
Lim et al.: *Sign and Basis Invariant Networks for Spectral Graph Representation Learning* (2022)
Zhang et al.: *Rethinking the expressive power of gnns via graph biconnectivity* (2023)] [2]

Spectral graph theory

- Eigenvalues/vector: $\mathbf{M} \cdot \mathbf{v} = \lambda \mathbf{v}$
- For adjacency matrices: Eigenvalues and eigenvectors of Laplacian $\mathbf{L}_G = \mathbf{D}_G - \mathbf{A}_G$

G



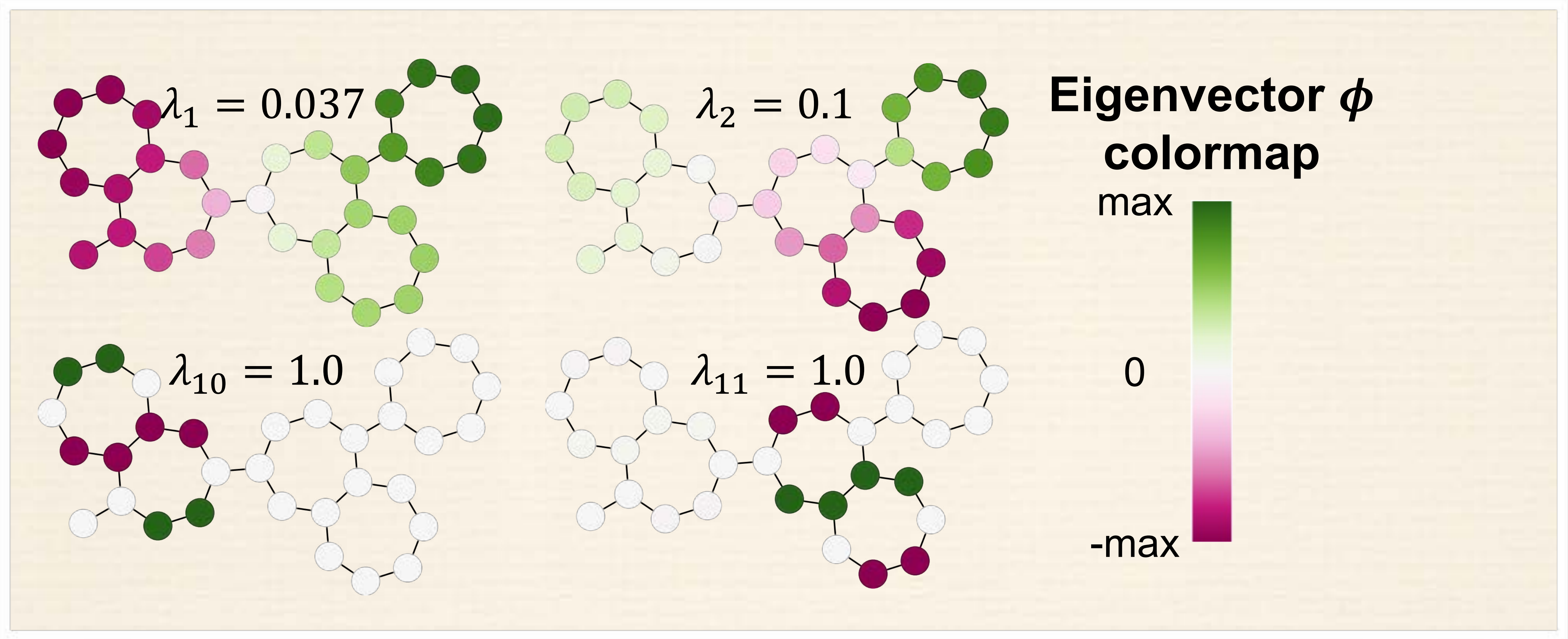
$$\underset{\mathbf{L}_G}{\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}} = \underset{\mathbf{D}_G}{\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}} - \underset{\mathbf{A}_G}{\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}}$$

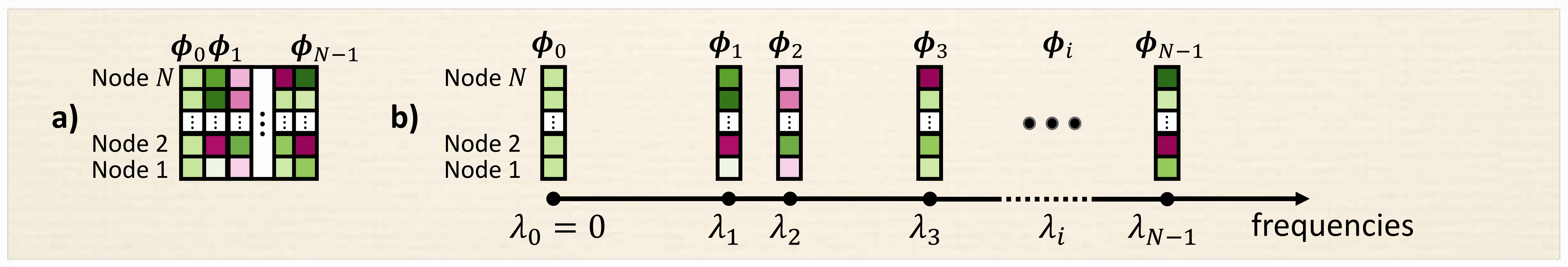
- Laplacian eigenvalues and vectors contain connectivity information
  - multiplicity 1st eigenvalue $\sim$ connected components.

Images: Wikipedia

Spectral MPNNs

Add eigenvectors as vertex features





Kreuzer et al.: *Rethinking graph transformers by spectral attention* (2021)

MPNNs+eig

- Add Laplacian eigenvectors (spectrum) as features.

eig=eigenvalue+eigenvectors

SpMPNNs

$$\xi^{(0)}(G,v) := \text{Hot-one encoding of label of vertex } v + \boxed{(\mathsf{eig}_1(v), \ldots, \mathsf{eig}_n(v))}$$

$$\xi^{(t)}(G,v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G,v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G,v), \xi^{(t)}(G,u), \xi^{(0)}, \boxed{(\mathsf{eig}_1(u), \ldots, \mathsf{eig}_n(u))} \mid u \in N_G(v)\}\}\big)\Big)$$

$$\rho(G) := \mathsf{Readout}\Big(\{\{\xi^{(L)}(G,v) \mid v \in V_G\}\}\Big)$$

- Ambiguity in eigenvector selection

- Not permutation invariant.

Kreuzer et al.: *Rethinking graph transformers by spectral attention* (2021)
Ying et al.: *Do transformers really perform bad for graph representation* (2021)

MPNNs+eig

- Add Laplacian eigenvectors (spectrum) as features.

eig=eigenvalue+eigenvectors

SpMPNNs

$$\xi^{(0)}(G,v) := \text{Hot-one encoding of label of vertex } v + (\mathsf{eig}_1(v), \ldots, \mathsf{eig}_n(v))$$

$$\xi^{(t)}(G,v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G,v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G,v), \xi^{(t)}(G,u), \xi^{(0)}, (\mathsf{eig}_1(u), \ldots, \mathsf{eig}_n(u)) \mid u \in N_G(v)\}\}\big)\Big)$$

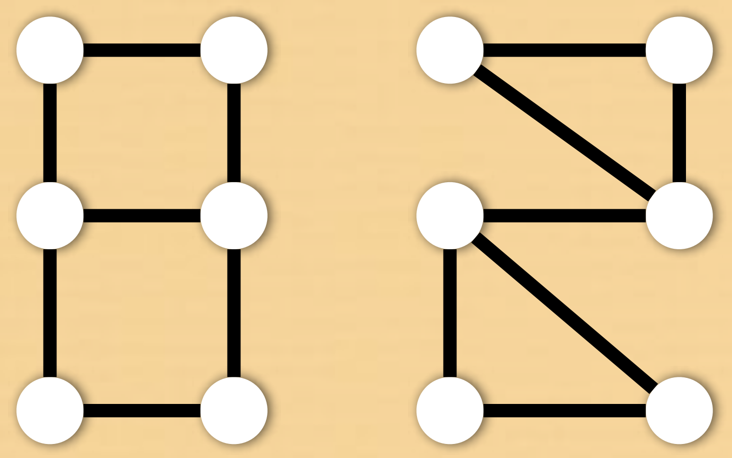
$$\rho(G) := \mathsf{Readout}\Big(\{\{\xi^{(L)}(G,v) \mid v \in V_G\}\}\Big)$$

- Ambiguity in eigenvector selection
- Not permutation invariant.

Kreuzer et al.: *Rethinking graph transformers by spectral attention* (2021)
Ying et al.: *Do transformers really perform bad for graph representation* (2021)

Expressive poser of MPNNs+eig

- Are as powerful as MPNNs with revised vertex labels



If eigenvectors assign unique vertex labels

See MPNNs + vertex ids

High expressive power

- Difficult to analyse.

Spectral invariant

$$\mathbf{A} = \sum_{\lambda} \lambda \mathbf{P}_{\lambda} \qquad \mathbf{P}_{\lambda} = \begin{pmatrix} p^{\lambda}_{11} & p^{\lambda}_{12} & \cdots & p^{\lambda}_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ p^{\lambda}_{n1} & p^{\lambda}_{n2} & \cdots & p^{\lambda}_{nn} \end{pmatrix}$$

Spectral invariant

$$v \mapsto \mathsf{specinv}(v) := (\lambda, p^{\lambda}_{vv}, \{\{p^{\lambda}_{vu} \mid u \in V_G\}\})_{\lambda \in \Lambda}$$

Cvetković et al.: *Eigenspaces of graphs* (1997)
M. Fürer: *On the power of combinatorial and spectral invariants* (2010)

Spectral invariant

$$\mathbf{A} = \sum_{\lambda} \lambda \mathbf{P}_{\lambda} \qquad \mathbf{P}_{\lambda} = \begin{pmatrix} p^{\lambda}_{11} & p^{\lambda}_{12} & \cdots & p^{\lambda}_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ p^{\lambda}_{n1} & p^{\lambda}_{n2} & \cdots & p^{\lambda}_{nn} \end{pmatrix}$$

Multiset

Spectral invariant

$$v \mapsto \mathsf{specinv}(v) := (\lambda, p^{\lambda}_{vv}, \{\{p^{\lambda}_{vu} \mid u \in V_G\}\})_{\lambda \in \Lambda}$$

Cvetković et al.: *Eigenspaces of graphs* (1997)
M. Fürer: *On the power of combinatorial and spectral invariants* (2010)

Spectral invariant

$$\mathbf{A} = \sum_{\lambda} \lambda \mathbf{P}_{\lambda} \qquad \mathbf{P}_{\lambda} = \begin{pmatrix} p^{\lambda}_{11} & p^{\lambda}_{12} & \cdots & p^{\lambda}_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ p^{\lambda}_{n1} & p^{\lambda}_{n2} & \cdots & p^{\lambda}_{nn} \end{pmatrix}$$

Multiset

Spectral invariant

$$v \mapsto \mathsf{specinv}(v) := (\lambda, p^{\lambda}_{vv}, \{\{p^{\lambda}_{vu} \mid u \in V_G\}\})_{\lambda \in \Lambda}$$

Graph properties

Number of length 3, 4, or 5 cycles, whether a graph is connected and the number of length k closed walks from any vertex to itself

Cvetković et al.: *Eigenspaces of graphs* (1997)
M. Fürer: *On the power of combinatorial and spectral invariants* (2010)

Spectral invariant

$$\mathbf{A} = \sum_{\lambda} \lambda \mathbf{P}_\lambda \qquad \mathbf{P}_\lambda = \begin{pmatrix} p^\lambda_{11} & p^\lambda_{12} & \cdots & p^\lambda_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ p^\lambda_{n1} & p^\lambda_{n2} & \cdots & p^\lambda_{nn} \end{pmatrix}$$

Multiset

Spectral invariant

$$v \mapsto \mathsf{specinv}(v) := (\lambda, p^\lambda_{vv}, \{\{p^\lambda_{vu} \mid u \in V_G\}\})_{\lambda \in \Lambda}$$

Graph properties

Number of length 3, 4, or 5 cycles, whether a graph is connected and the number of length k closed walks from any vertex to itself

Beyond 1-WL/Colour Refinement

Cvetković et al.: *Eigenspaces of graphs* (1997)
M. Fürer: *On the power of combinatorial and spectral invariants* (2010)

SpecMPNN

Spectral invariant

$$v \mapsto \mathsf{specinv}(v) := (\lambda, p^{\lambda}_{vv}, \{\{p^{\lambda}_{vu} \mid u \in V_G\}\})_{\lambda \in \Lambda}$$

Variation used in Signet and BasisNet → 2-WL bound

Can be using combination with any MPNN

Theorem (Seppelt and Rattan (2023)

specMPNN bounded in power by (1,1)-WL and strictly lower than 2-WL

Lim et al.: *Sign and Basis Invariant Networks for Spectral Graph Representation Learning* (2022)
G. Rattan and T. Seppelt: *Weisfeiler-Leman and Graph Spectra* (2023)

SpecMPNN

Spectral invariant

$$v \mapsto \mathsf{specinv}(v) := \left(\lambda, p^{\lambda}_{vv}, \{\{p^{\lambda}_{vu} \mid u \in V_G\}\}\right)_{\lambda \in \Lambda}$$

Variation used in Signet and BasisNet → 2-WL bound

Can be using combination with any MPNN

Theorem (Seppelt and Rattan (2023)

specMPNN bounded in power by (1,1)-WL and strictly lower than 2-WL

We discuss these WL's later

Lim et al.: *Sign and Basis Invariant Networks for Spectral Graph Representation Learning* (2022)
G. Rattan and T. Seppelt: *Weisfeiler-Leman and Graph Spectra* (2023)

1-rMPNN+
randomness clouds
k-rMPNN+
∞ – rMPNN+
$\mathscr{P}^\star$-MPNNs
$\mathscr{P}$-SGNs
$\mathscr{P}$-MPNNs
2-WL
MPNNs+eig
Signnet
(1,1)-WL
specMPNNs
Signnet?
MPNN+s
1-WL
Colour Refinement
GCN
MPNNs
GIN
$\mathscr{H}$
Message passing
Expressiveness
Complexity
not invariant

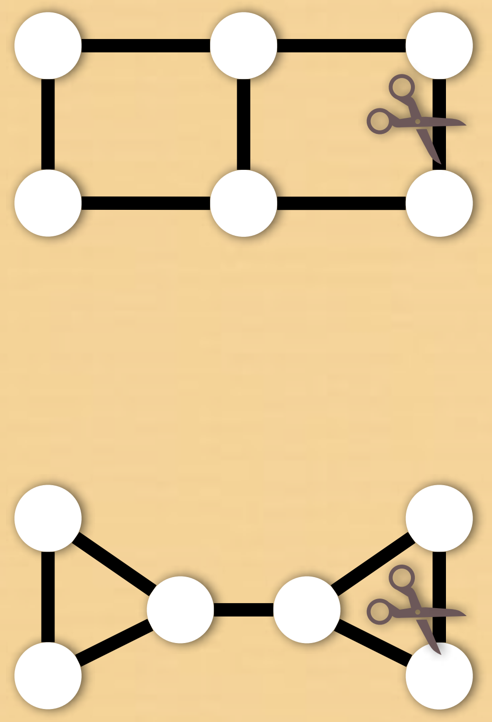
Questions?

Subgraph GNNs

Turning one graph into many

General idea

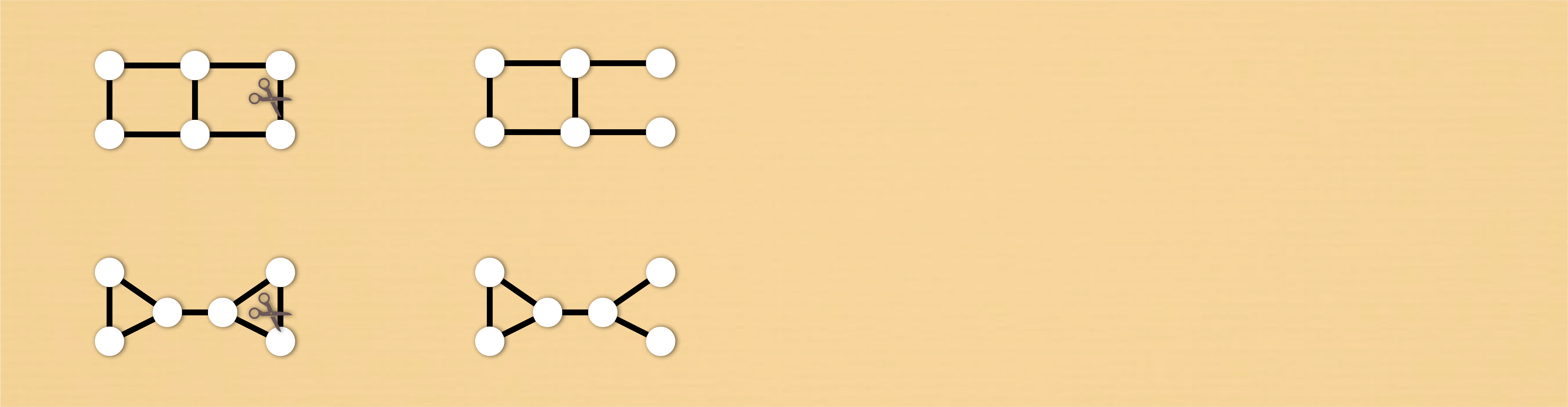
- Colour refinement equivalent graphs may contain colour refinement inequivalent subgraphs.



- View graphs as a collection of subgraphs then run MPNN

General idea

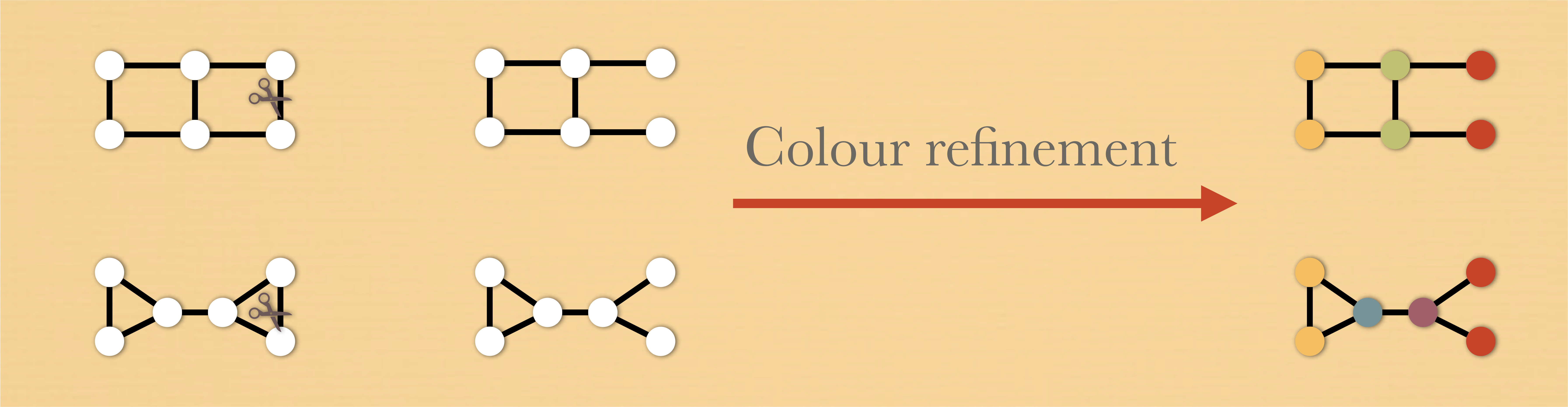
- Colour refinement equivalent graphs may contain colour refinement inequivalent subgraphs.



- View graphs as a collection of subgraphs then run MPNN

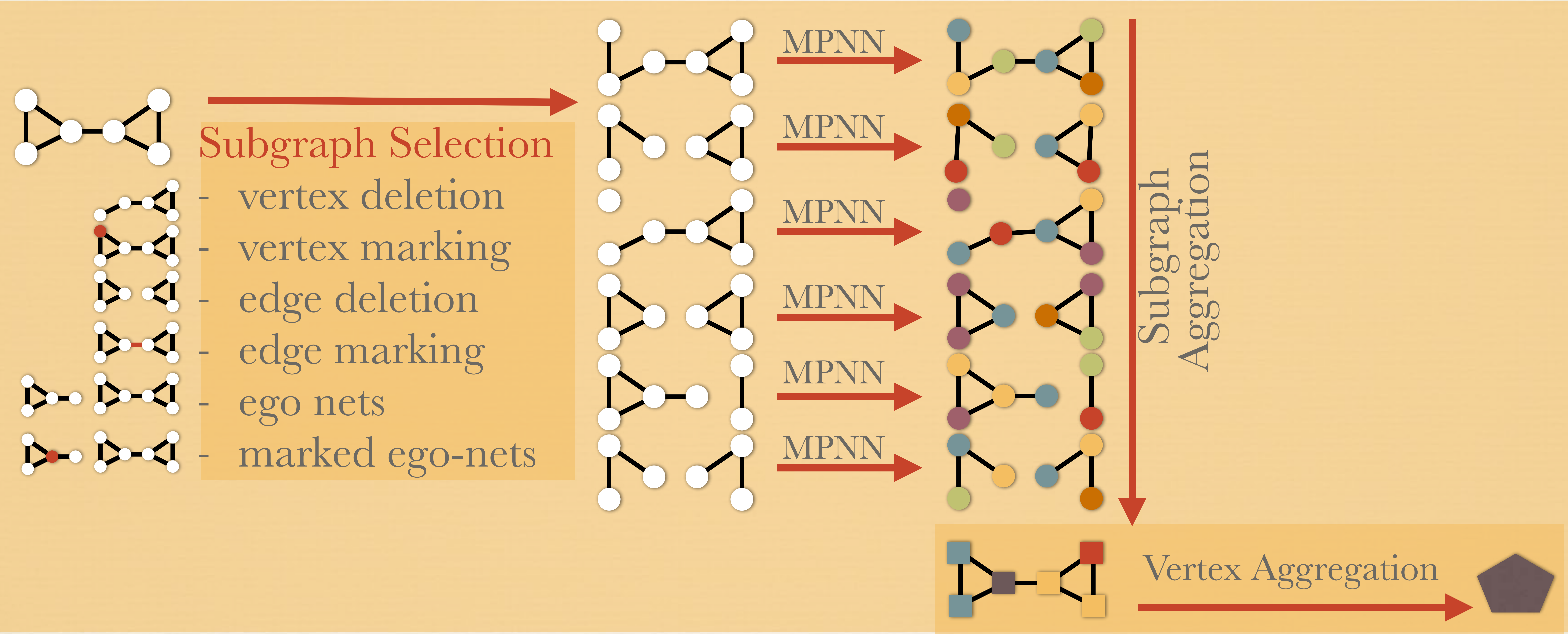
General idea

- Colour refinement equivalent graphs may contain colour refinement inequivalent subgraphs.

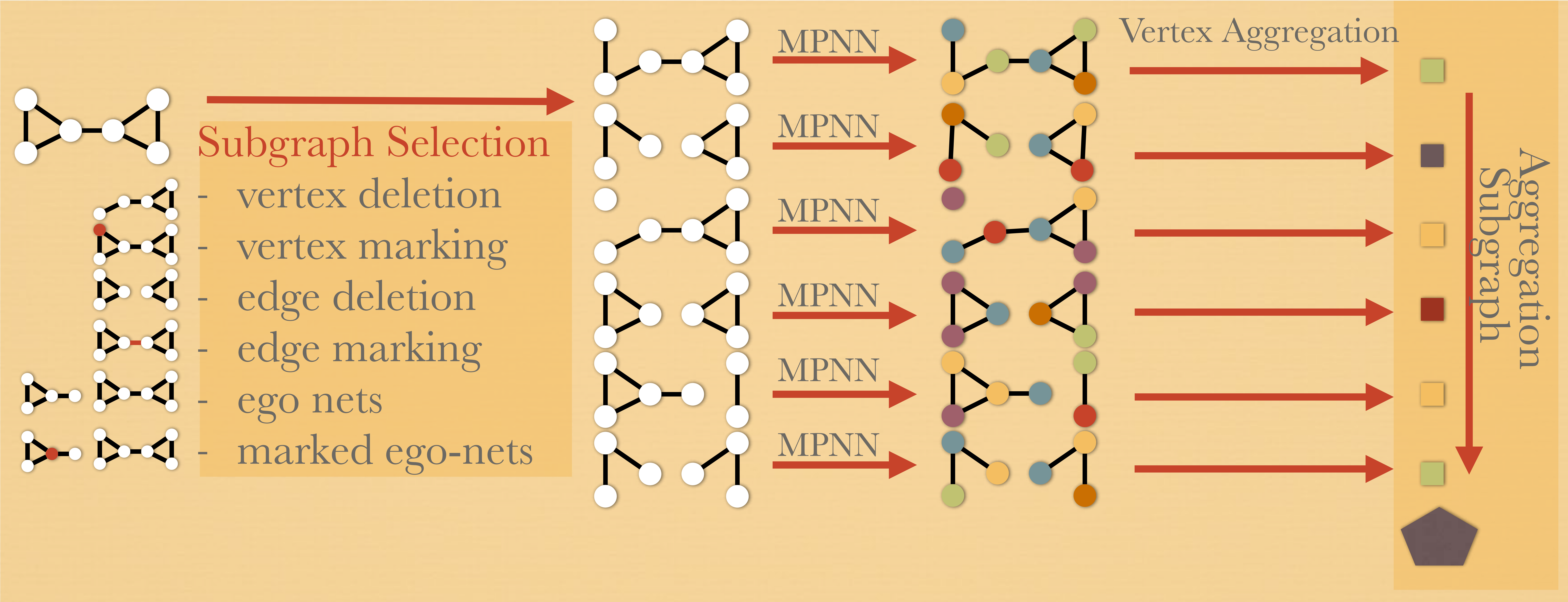


- View graphs as a collection of subgraphs then run MPNN

Subgraph↦Vertex Aggregation



Vertex↦Subgraph Aggregation



The subgraph GNN “wave”

Vertex↦Subgraph Aggregation

Subgraph↦Vertex Aggregation

k-OSAN^T

Reconstruction GNN

DS-GNN

DSS-GNN

GNN-AK

k-OSAN

NGNN

ID-GNN

DropoutGNN

All provably more expressive than MPNNs*

Bevilacqua et al: *Equivariant subgraph aggregation network* (2022)
Cotta et al.: *Reconstruction for powerful graph representations* (2021)
Bevilacqua et al.: *Understanding and extending subgraph GNNs by rethinking their symmetries* (2022)
Huang et al.: *Boosting the cycle counting power of graph neural networks with I2-GNNs* (2022)
Papp et al.: *DropGNN: Random dropouts increase the expressiveness of graph neural networks.* (2021)
Qian et al.: *Ordered subgraph aggregation networks.* (2022)
You et al.: *Identity-aware graph neural networks.* (2021)
Zhang and P. Li. *Nested graph neural networks* (2021)
Zhao et al.: *From stars to subgraphs: Uplifting any GNN with local structure awareness* (2022)

The subgraph GNN "wave"

Vertex↦Subgraph Aggregation

Subgraph↦Vertex Aggregation

k-OSAN^T

Reconstruction GNN

DS-GNN

DSS-GNN

GNN-AK

k-OSAN

NGNN

ID-GNN

DropoutGNN

All provably more expressive than MPNNs*

Bevilacqua et al: *Equivariant subgraph aggregation network* (2022)
Cotta et al.: *Reconstruction for powerful graph representations* (2021)
Bevilacqua et al.: *Understanding and extending subgraph GNNs by rethinking their symmetries* (2022)
Huang et al.: *Boosting the cycle counting power of graph neural networks with I2-GNNs* (2022)
Papp et al.: *DropGNN: Random dropouts increase the expressiveness of graph neural networks*. (2021)
Qian et al.: *Ordered subgraph aggregation networks.* (2022)
You et al.: *Identity-aware graph neural networks*. (2021)
Zhang and P. Li. *Nested graph neural networks* (2021)
Zhao et al.: *From stars to subgraphs: Uplifting any GNN with local structure awareness* (2022)

Selection policies

DS-GNN
- vertex deletion
- edge deletion
- ego nets
- marked ego-nets

ID-GNNs
- marked ego-nets

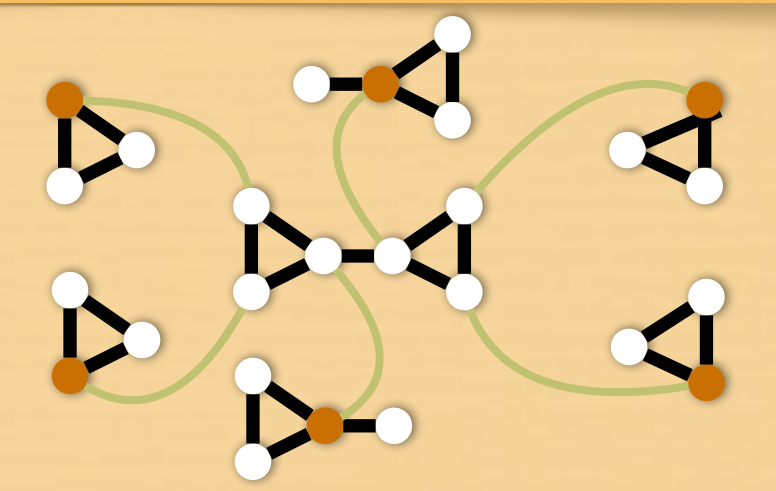
GNNs-AK
- ego-nets

k-OSAN
- size k subgraph marking

Rec-GNN
- k-vertex deletion

NGNN
- ego-nets

Popular/effective: ego-nets



General Subgraph MPNNs

- We discuss an extension of MPNNs called Ordered Subgraph Aggregation Networks
- General enough to capture most existing methods*
- Theoretical results on expressive power of OSANs translate directly to these methods.

*Except for (possibly) Bevilacqua et al: *Equivariant subgraph aggregation network* (2022)
Qian et al.: *Ordered subgraph aggregation networks.* (2022)

k-OSAN

Initialisation:

Selection of k tuple of vertices $\mathbf{g}$

$$\pi(v, \mathbf{g}) := \mathsf{UPD}_\pi(\text{type of } \mathbf{g}, v)$$

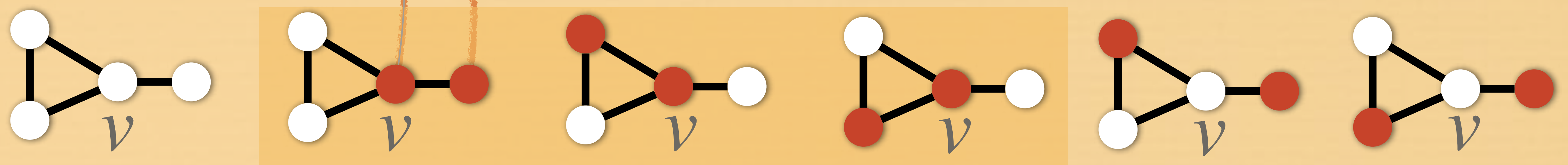
Induced subgraph

Initial labels

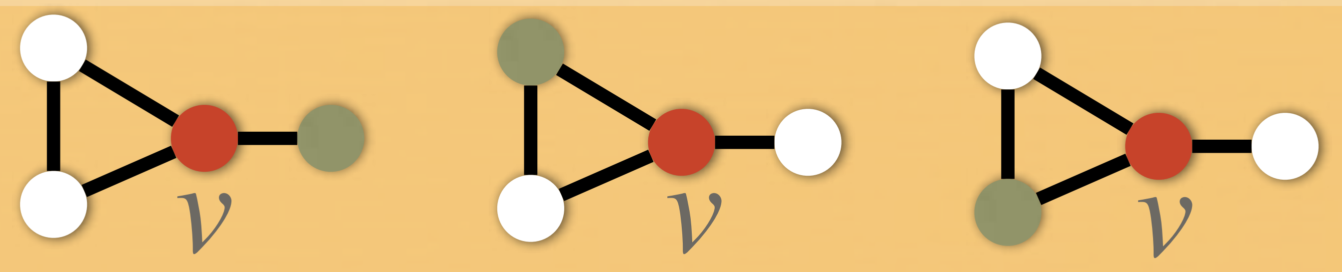
$$\xi^{(0)}(v, \mathbf{g}) := \mathsf{UPD}(\text{type of } \mathbf{g}, v)$$

Learnable function (MLP)

Only edges adjacent to v



Label them differently

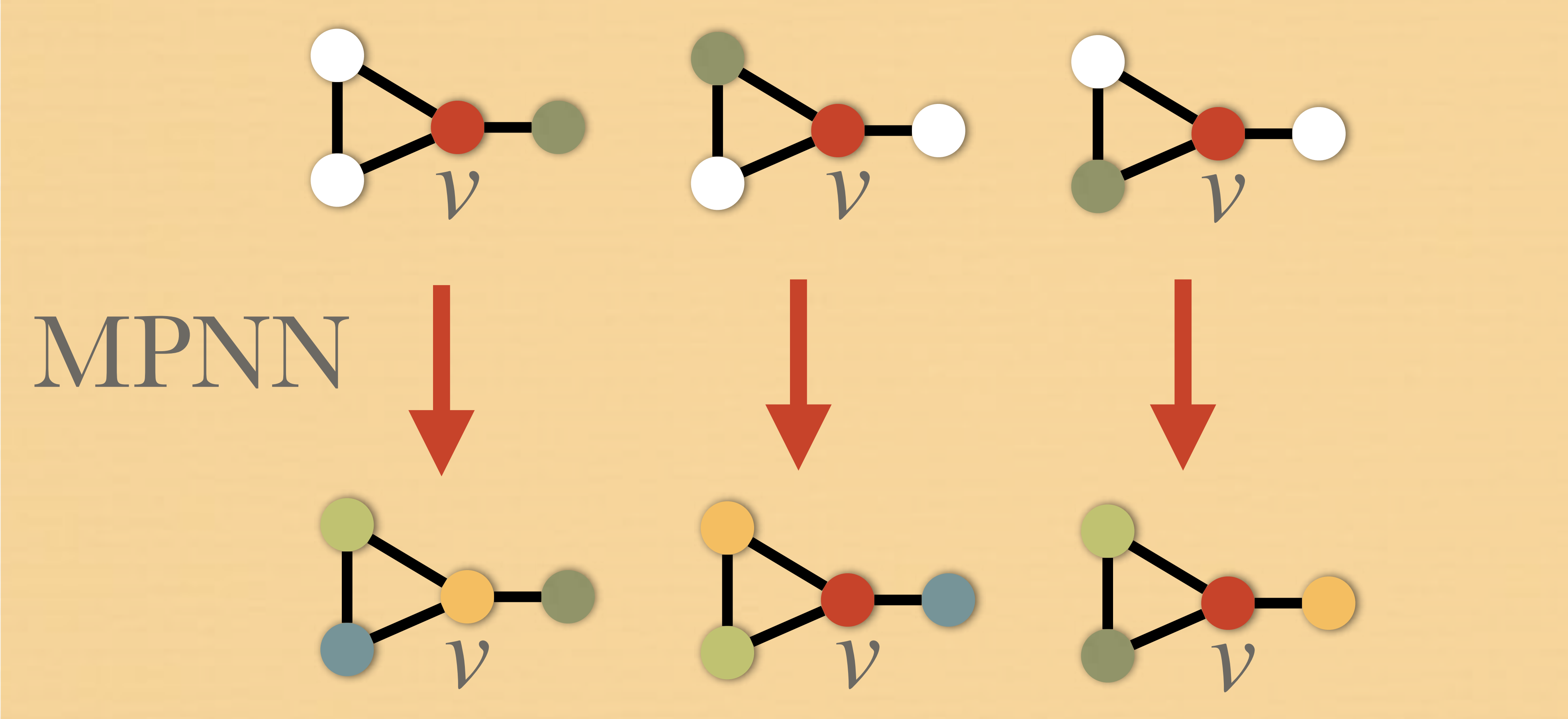


Qian et al.: *Ordered subgraph aggregation networks.* (2022)

k-OSAN

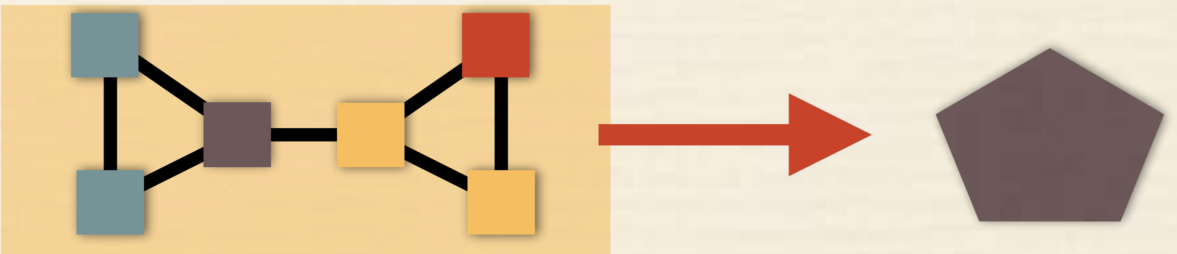
Iteration t: run MPNN for each $\mathbf{g}$

$$\xi^{(t)}(v, \mathbf{g}) := \mathsf{UPD}^{(t)}(\xi^{(t)}(v, \mathbf{g}), \mathsf{AGG}^{(t)}(\{\{\xi^{(t)}(u, \mathbf{g}) \mid u \in V_G \text{ or } N_G(v)\}\})$$



Subgraph↦vertex Aggregation

$$\xi(v) := \mathsf{AGG}(\{\{\xi^{(L)}(v, \mathbf{g}) \mid \pi(v, \mathbf{g}) \neq 0\}\})$$



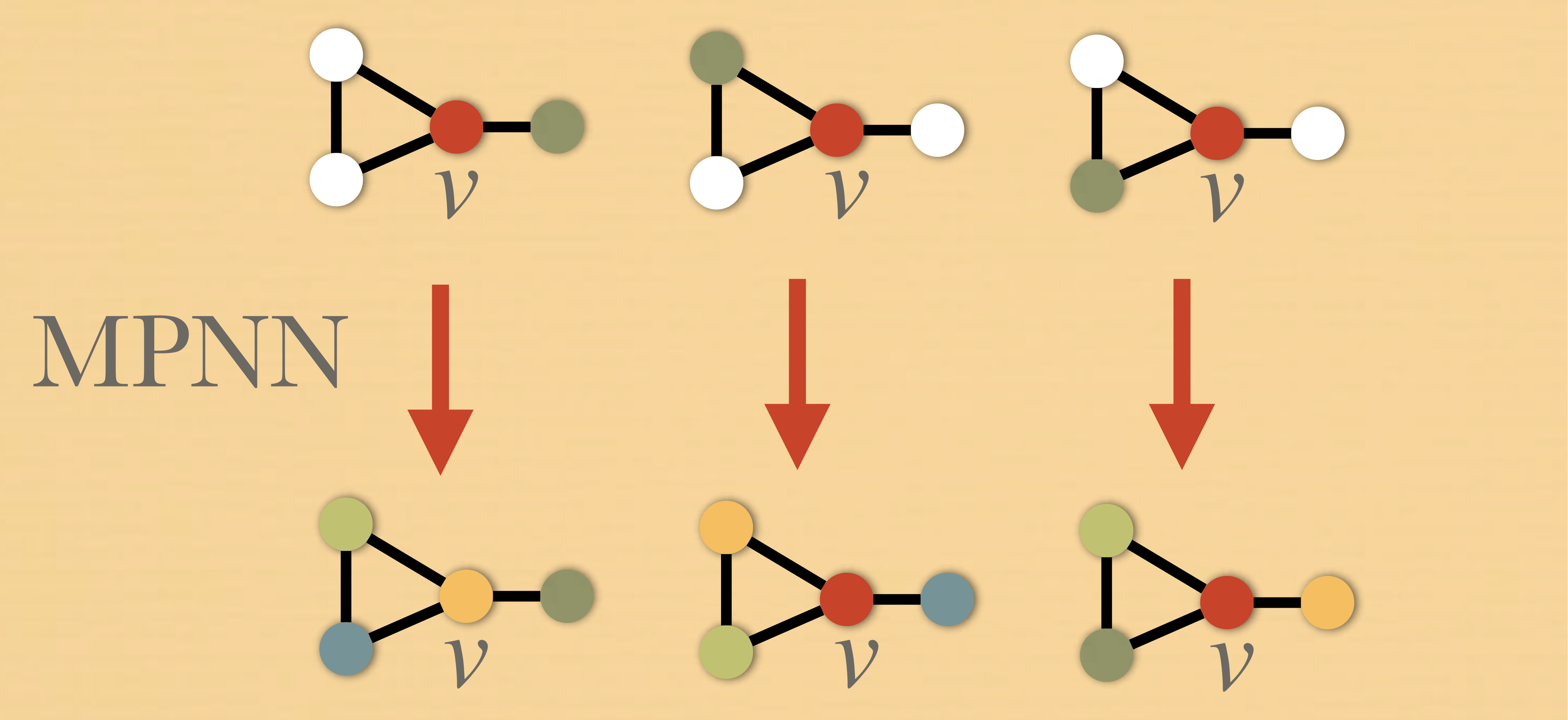
Selection policy

Qian et al.: *Ordered subgraph aggregation networks.* (2022)

k-OSAN^T

Iteration t: run MPNN for each $\mathbf{g}$

$$\xi^{(t)}(v, \mathbf{g}) := \mathsf{UPD}^{(t)}(\xi^{(t)}(v, \mathbf{g}), \mathsf{AGG}^{(t)}(\{\{\xi^{(t)}(u, \mathbf{g}) \mid u \in V_G \text{ or } N_G(v)\}\}))$$



Vertex↦Subgraph aggregation

$$\xi(\mathbf{g}) := \mathsf{AGG}(\{\{\xi^{(L)}(v, \mathbf{g}) \mid \pi(v, \mathbf{g}) \neq 0, v \in V_G\}\})$$

Qian et al.: *Ordered subgraph aggregation networks.* (2022)

k-OSAN

Theorem (Qian et al. 2022)

- k-OSANs and k-OSANs^t encompass *almost all* subgraph methods with selection policy involving k vertices.
- *Strictly bounded* in expressive power by *(k+1)-WL*
- *Incomparable* to k-WL.

k=2

- if 2-WL cannot distinguish graphs, then neither can 1-OSANs
- 2-WL can distinguish more graphs than 1-OSANs
- There exists graphs than can be distinguished by 1-OSANs but not by MPNNs, and vice versa, there exists graphs that can be distinguished by MPNNs but not by 1-OSAS

Qian et al.: *Ordered subgraph aggregation networks.* (2022)

Subgraph GNNs

- Can always ensure to be strictly more expressive than MPNNs by including original graph in batch.

- Tractability only when easy subgraph policies are used, i.e., leading to a small number (linear) of subgraphs.

- Seems a good balance between complexity and expressiveness

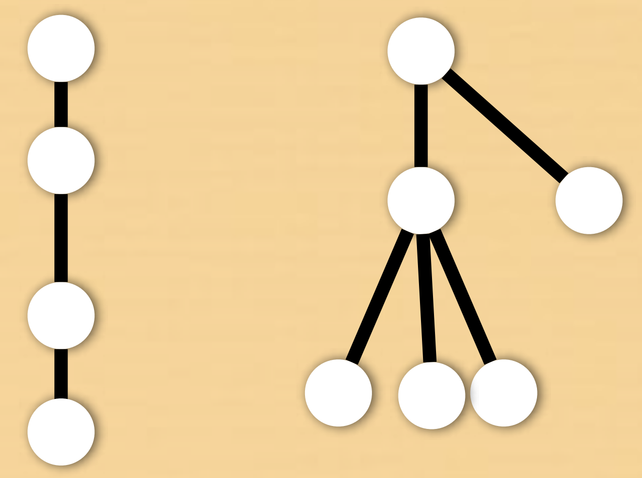

1-rMPNN+
randomness clouds
k-rMPNN+
∞ – rMPNN+
k-WL
(K-1)-OSAN
𝒫⋆-MPNNs
𝒫-SGNs
𝒫-MPNNs
2-WL
MPNNs+eig
Signnet
(1,1)-WL
DSS-GNN
specMPNNs
DS-GNN/1-OSAN/1-OSAN'
Signnet?
MPNN+s
GNN-AK
1-WL
Colour Refinement
NGNN
GCN
MPNNs
ID-GNN
ℋ
GIN
DropoutGNN
Message passing
Expressiveness
Complexity
not invariant

Characterisation $\rho(k\text{-OSAN})$

- To our knowledge no characterisation of the expressive power of subgraph GNNs (and k-OSANs in particular) in terms of homomorphism counts is known.
- An exception are the 1-OSANs.

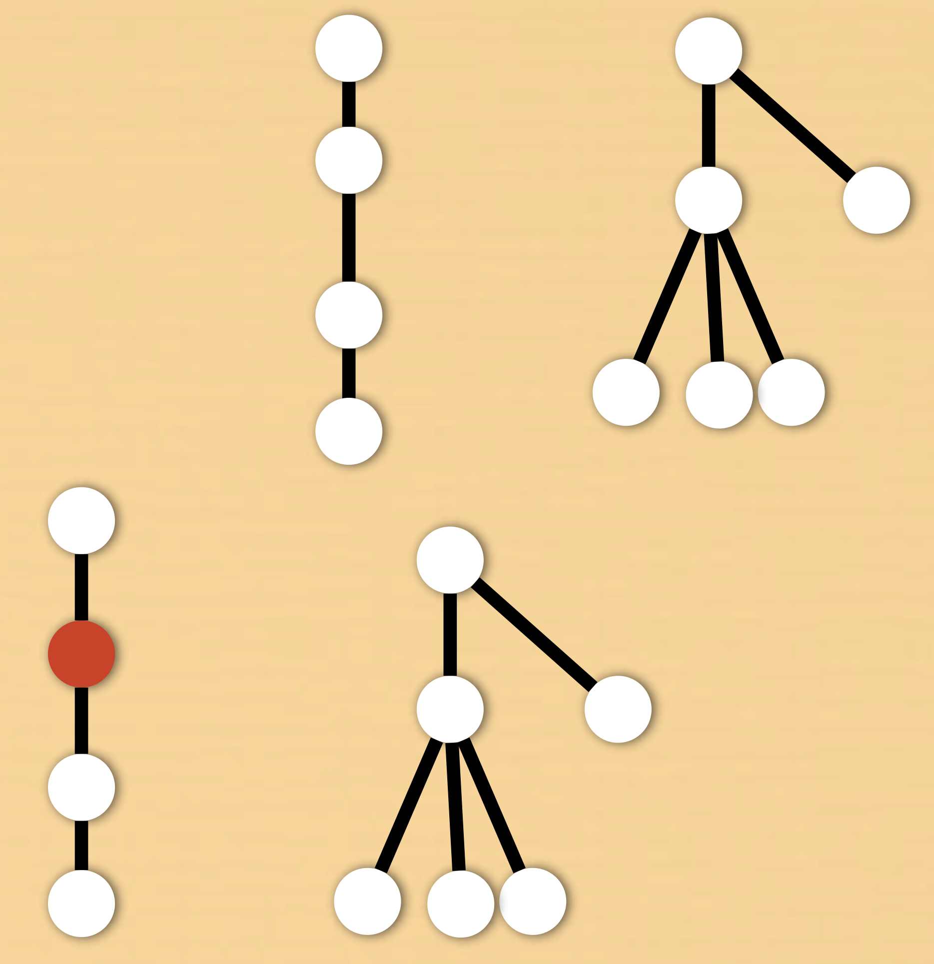
Characterisation ρ(1-OSAN)

- Let $\mathscr{F}$ be the class of all forests (collection of trees)
- Let $\mathscr{F}^+$ be collection of graphs obtained by
  - Taking forest $F \in \mathscr{F}$
  - Taking set $\emptyset \neq B \subseteq V_F$ of vertices
  - Contracting all vertices in B to a single vertex (removing loops and multi edges).



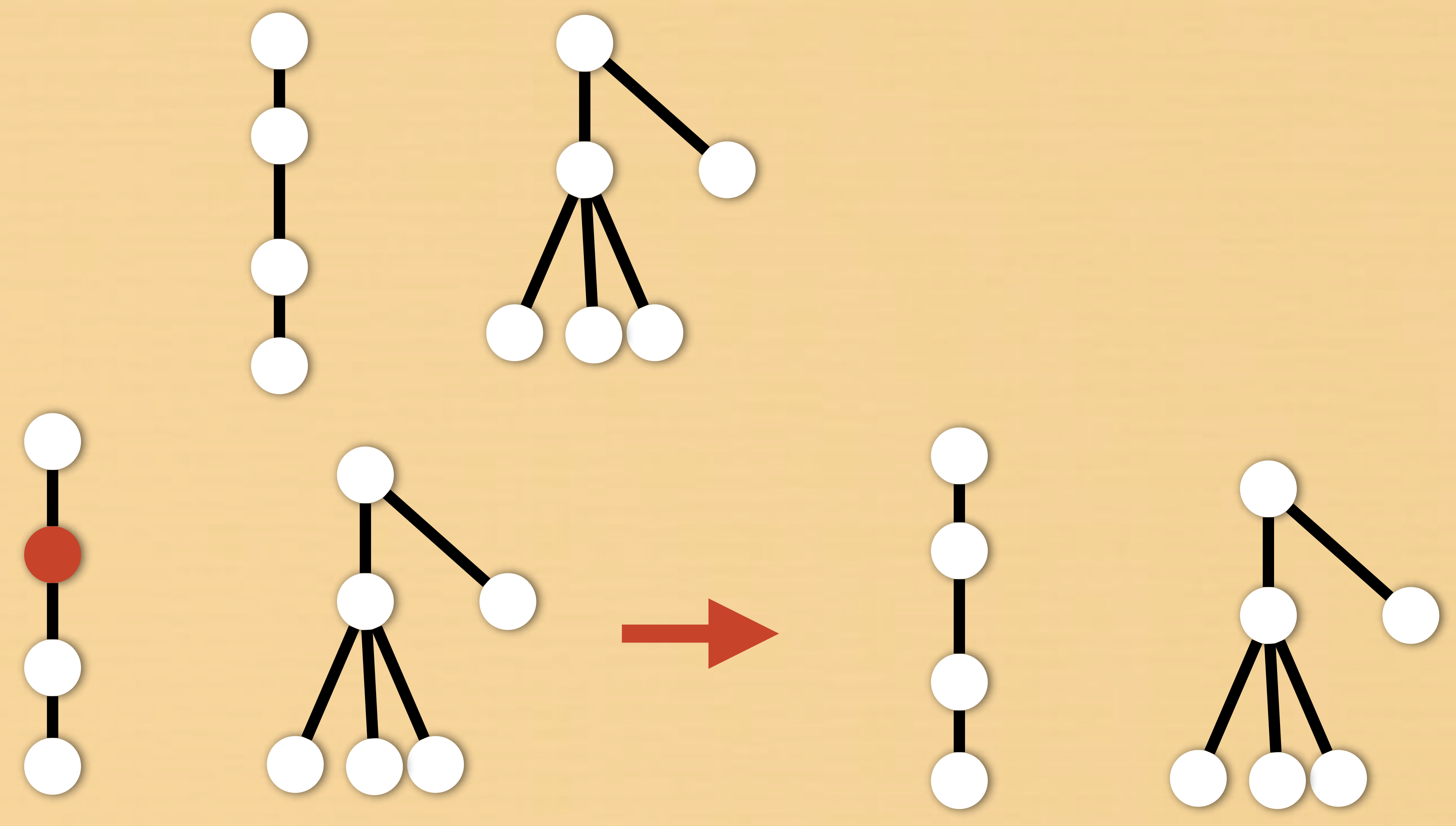
Characterisation ρ(1-OSAN)

- Let $\mathscr{F}$ be the class of all forests (collection of trees)
- Let $\mathscr{F}^+$ be collection of graphs obtained by
  - Taking forest $F \in \mathscr{F}$
  - Taking set $\emptyset \neq B \subseteq V_F$ of vertices
  - Contracting all vertices in B to a single vertex (removing loops and multi edges).



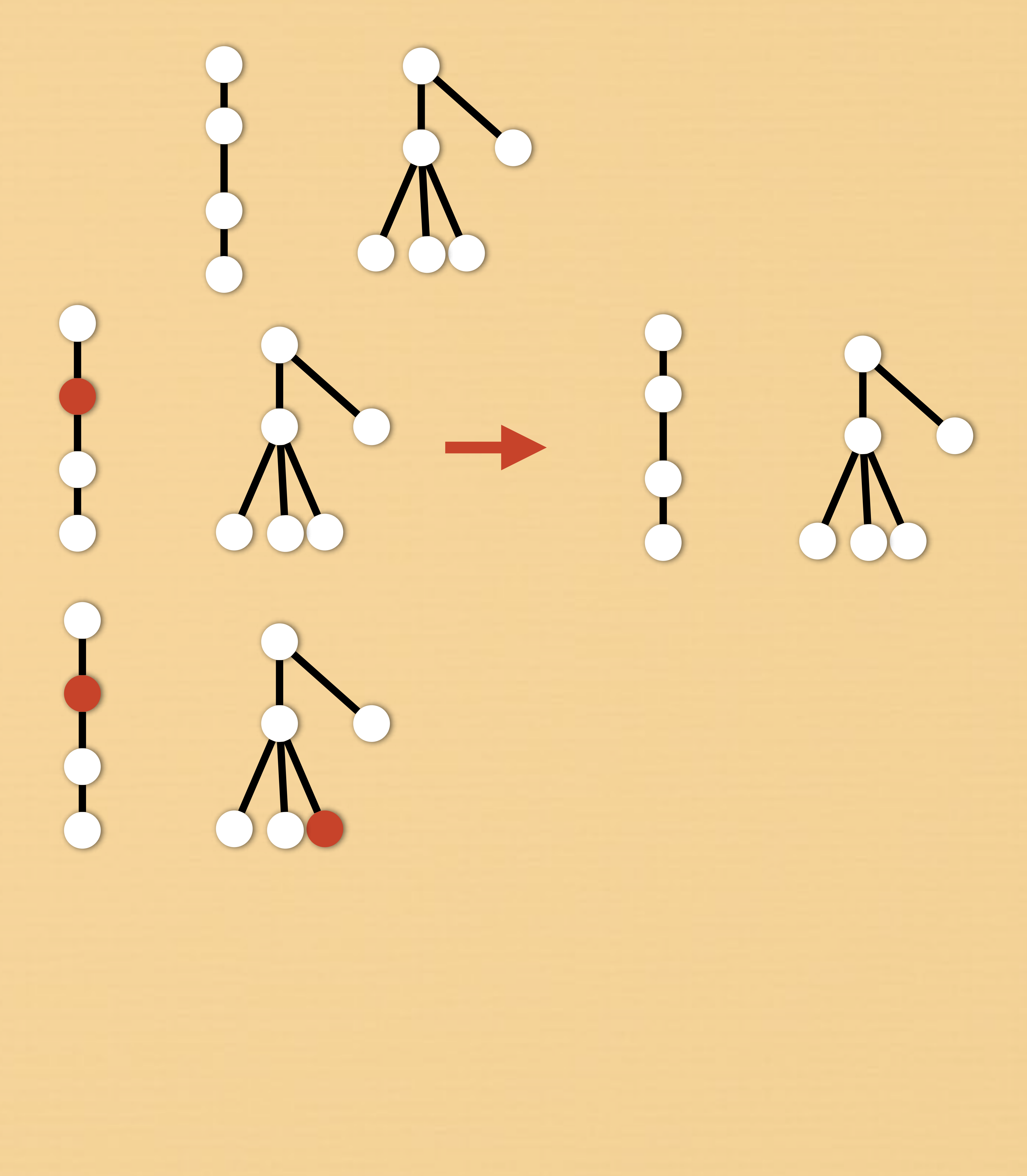
Characterisation ρ(1-OSAN)

- Let $\mathscr{F}$ be the class of all forests (collection of trees)
- Let $\mathscr{F}^+$ be collection of graphs obtained by
  - Taking forest $F \in \mathscr{F}$
  - Taking set $\varnothing \neq B \subseteq V_F$ of vertices
  - Contracting all vertices in B to a single vertex (removing loops and multi edges).



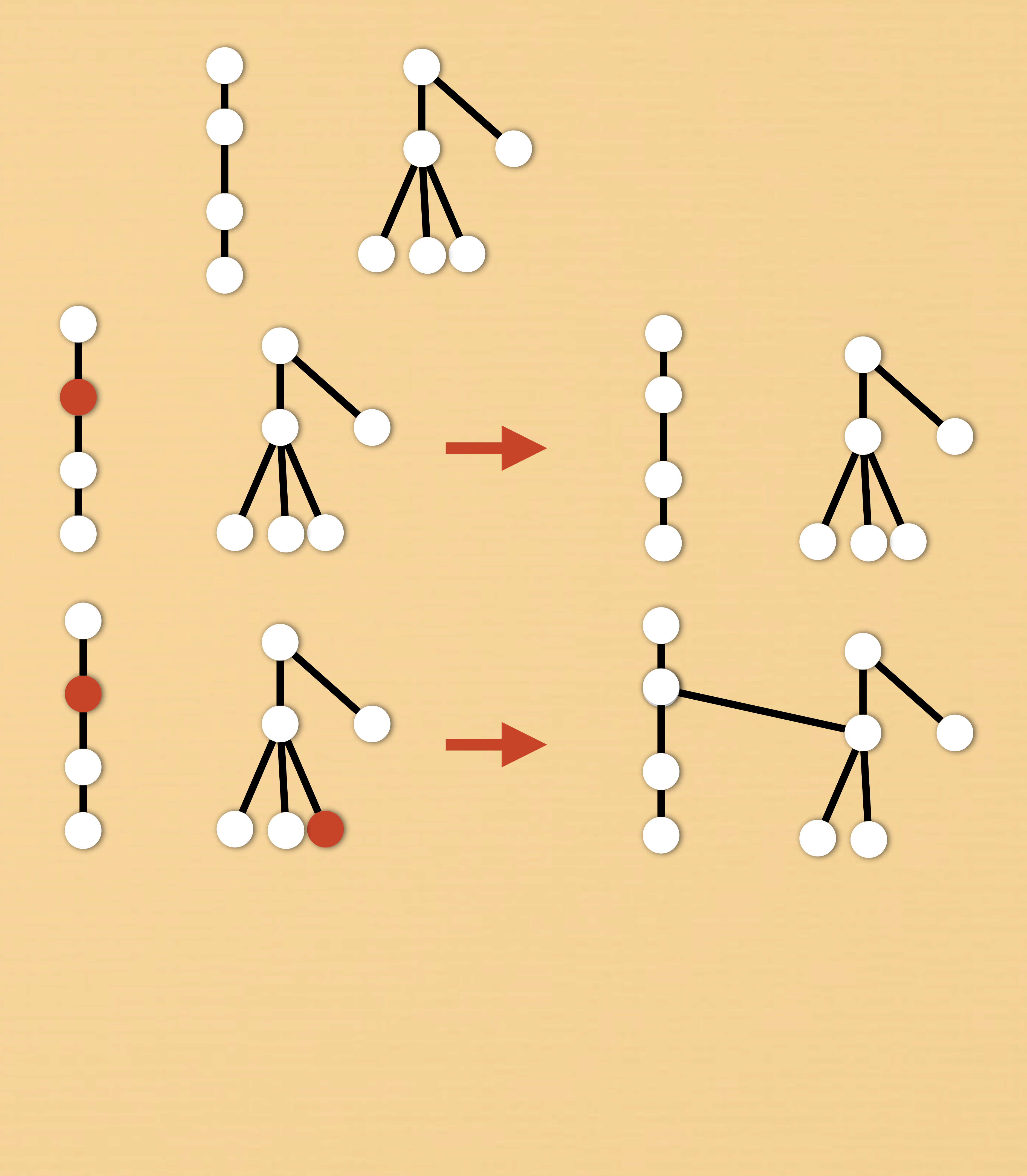
Characterisation ρ(1-OSAN)

- Let $\mathscr{F}$ be the class of all forests (collection of trees)
- Let $\mathscr{F}^+$ be collection of graphs obtained by
    - Taking forest $F \in \mathscr{F}$
    - Taking set $\emptyset \neq B \subseteq V_F$ of vertices
    - Contracting all vertices in B to a single vertex (removing loops and multi edges).



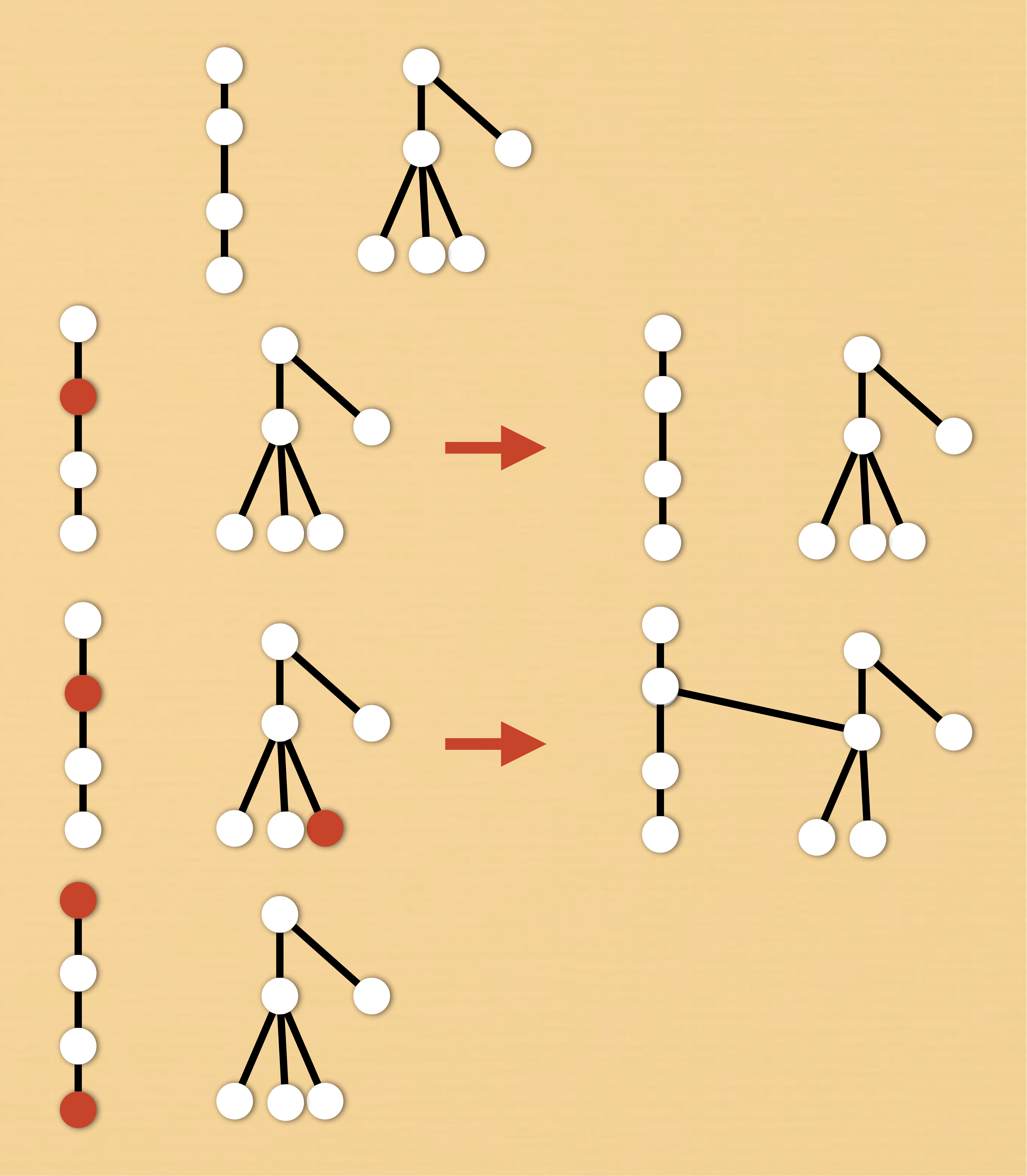
Characterisation ρ(1-OSAN)

- Let $\mathcal{F}$ be the class of all forests (collection of trees)
- Let $\mathcal{F}^+$ be collection of graphs obtained by
  - Taking forest $F \in \mathcal{F}$
  - Taking set $\emptyset \neq B \subseteq V_F$ of vertices
  - Contracting all vertices in B to a single vertex (removing loops and multi edges).



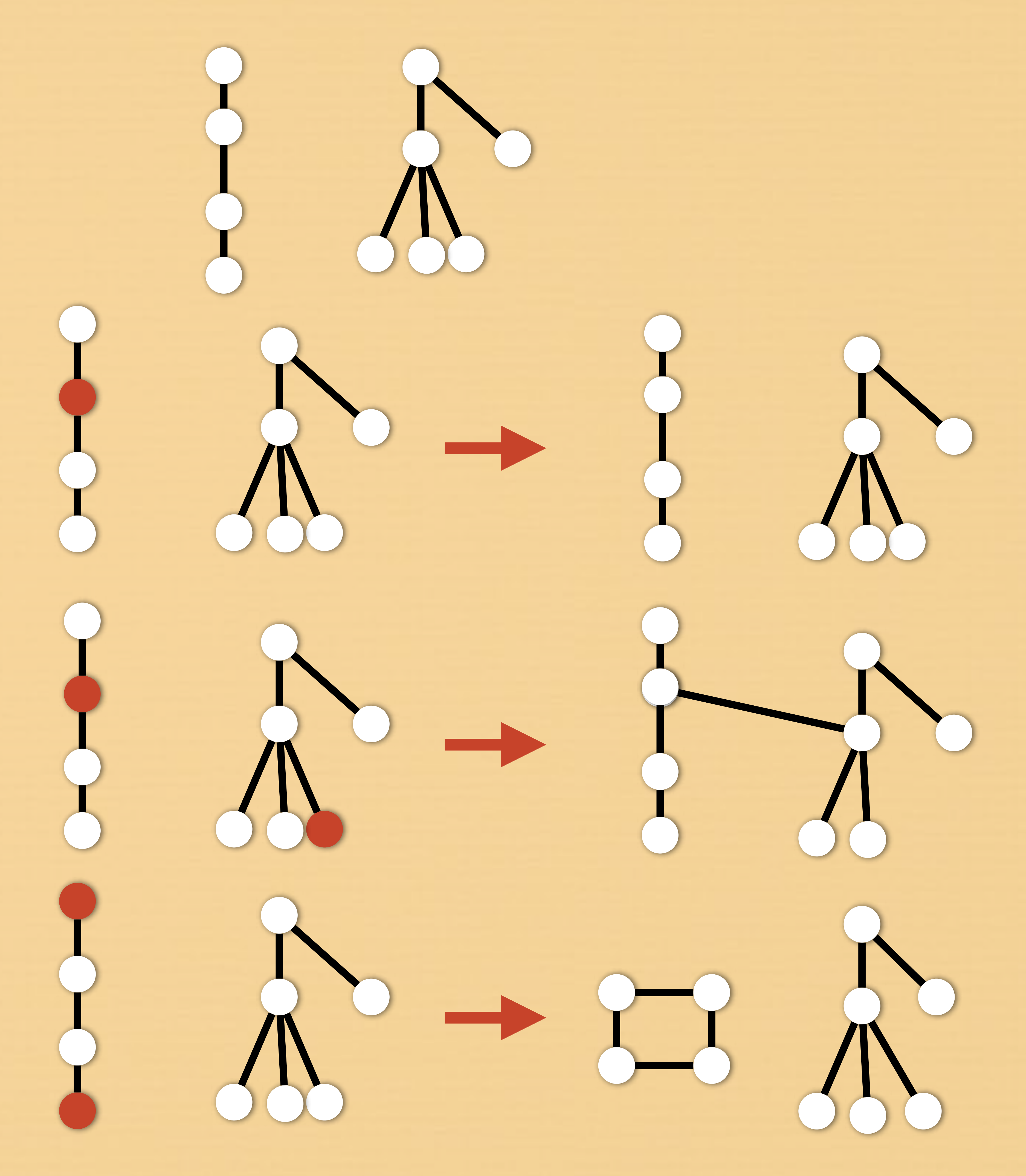
Characterisation ρ(1-OSAN)

- Let $\mathscr{F}$ be the class of all forests (collection of trees)
- Let $\mathscr{F}^+$ be collection of graphs obtained by
  - Taking forest $F \in \mathscr{F}$
  - Taking set $\emptyset \neq B \subseteq V_F$ of vertices
  - Contracting all vertices in B to a single vertex (removing loops and multi edges).



Characterisation ρ(1-OSAN)

- Let $\mathscr{F}$ be the class of all forests (collection of trees)
- Let $\mathscr{F}^+$ be collection of graphs obtained by
  - Taking forest $F \in \mathscr{F}$
  - Taking set $\emptyset \neq B \subseteq V_F$ of vertices
  - Contracting all vertices in B to a single vertex (removing loops and multi edges).



Characterisation ρ(1-OSAN)

- Note: $\mathscr{F}^+$ contains all trees, but also cycles etc.
- Note: treewidth of elements in $\mathscr{F}^+$ is at most two.

Theorem (Seppelt & Rattan, 2023)

$\mathsf{hom}(F, G) = \mathsf{hom}(F, H)$ for all $F \in \mathscr{F}^+$
if and only if
no 1-OSAN can distinguish G from H.

- 1-OSANs (and also ID-aware GNNs, …) have the ability to detect cycles, etc.

Lim et al.: *Sign and Basis Invariant Networks for Spectral Graph Representation Learning* (2022)
G. Rattan and T. Seppelt: *Weisfeiler-Leman and Graph Spectra* (2023)


1-rMPNN+
randomness clouds
k-rMPNN+
∞ – rMPNN+
k-WL
(K-1)-OSAN
𝒫*-MPNNs
𝒫-SGNs
𝒫-MPNNs
2-WL
MPNNs+eig
Signnet
(1,1)-WL
DSS-GNN
specMPNNs
DS-GNN/1-OSAN/1-OSAN'
Signnet?
MPNN+s
GNN-AK
1-WL
Colour Refinement
NGNN
GCN
MPNNs
ID-GNN
ℋ
GIN
Message passing
DropoutGNN
Expressiveness
Complexity
not invariant

Questions?

K-dimensional Weisfeiler-Leman

Boosting expressive power by higher-order message-passing

Motivation

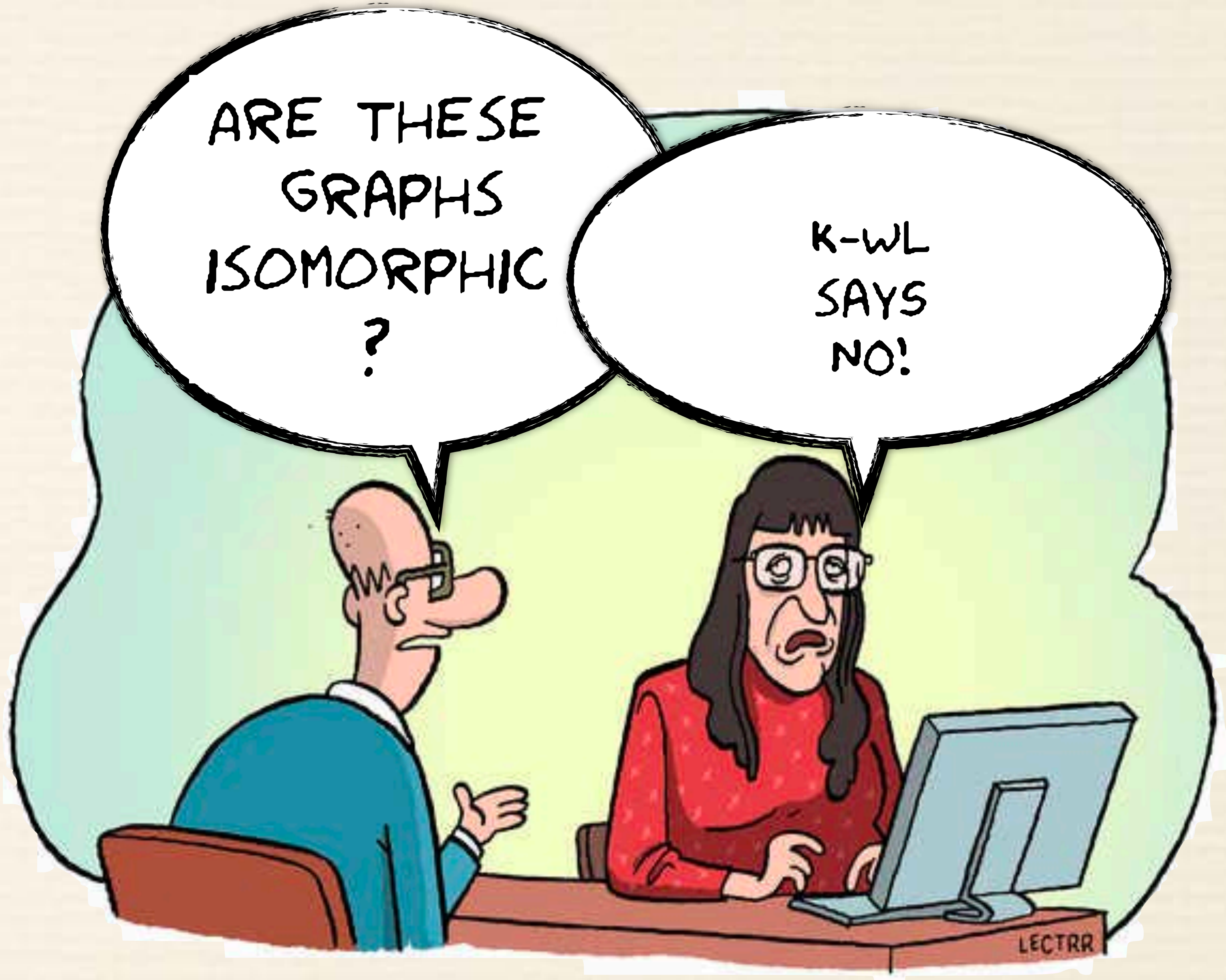
- We have seen that many graph embedding methods are bounded in expressive power by 1-WL or colour refinement
- To go beyond this, one can manually add more expressive features.
- In the theoretical computer science community, however, higher-order version of 1-WL have been studied for a long time.
- Why not use these to build more powerful embedding methods?

More powerful heuristic

Apply heuristic on G and H: If Heuristic say "no" then $G \ncong H$, otherwise we do not know.

$G \cong H$?

Colour refinement → No → $G \ncong H$



More powerful heuristic

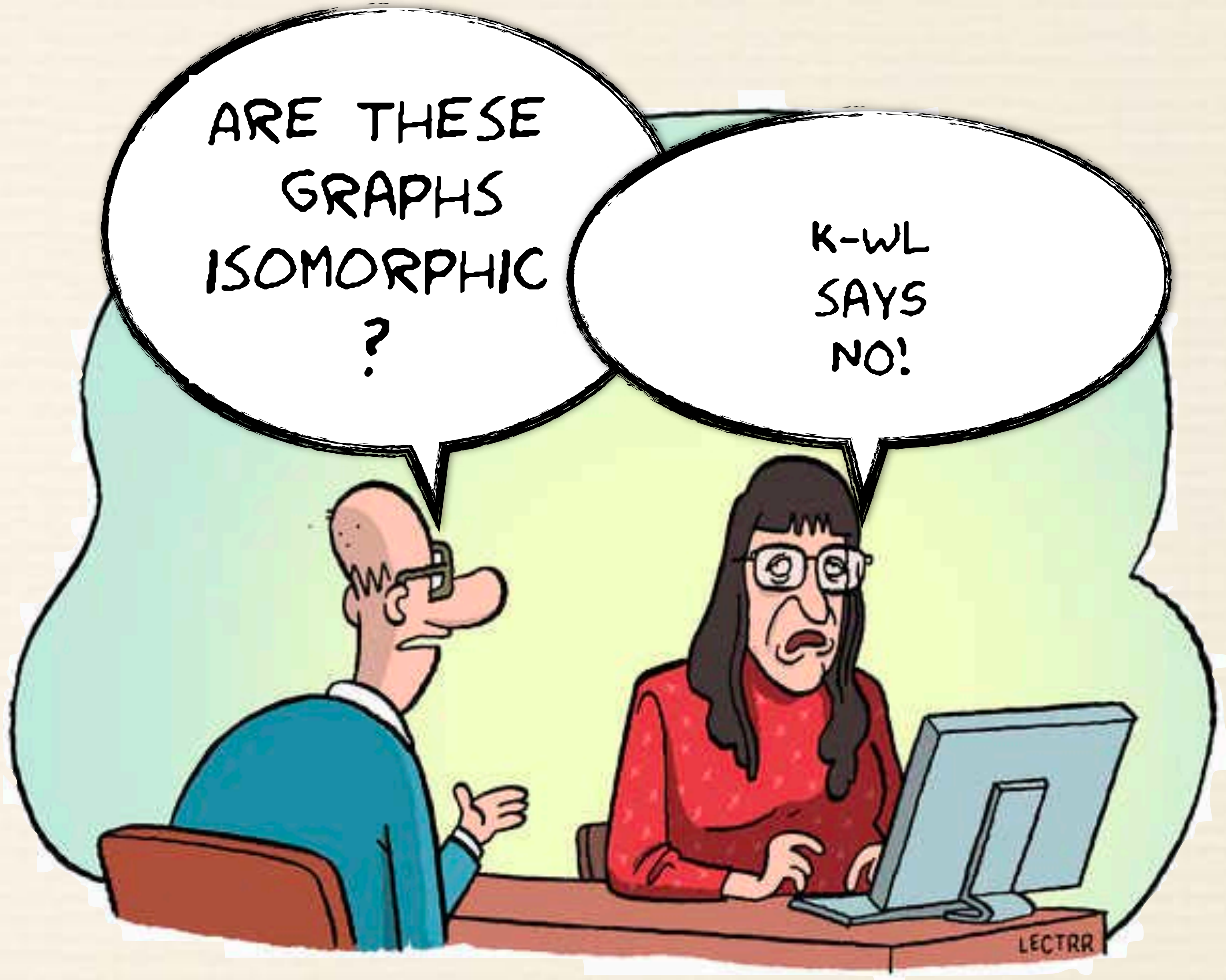
Apply heuristic on G and H: If Heuristic say "no" then $G \ncong H$, otherwise we do not know.

$G \cong H$?

Colour refinement → No → $G \ncong H$

↓

1-WL → No → $G \ncong H$



More powerful heuristic

Apply heuristic on G and H: If Heuristic say "no" then $G \ncong H$, otherwise we do not know.

$G \cong H$?

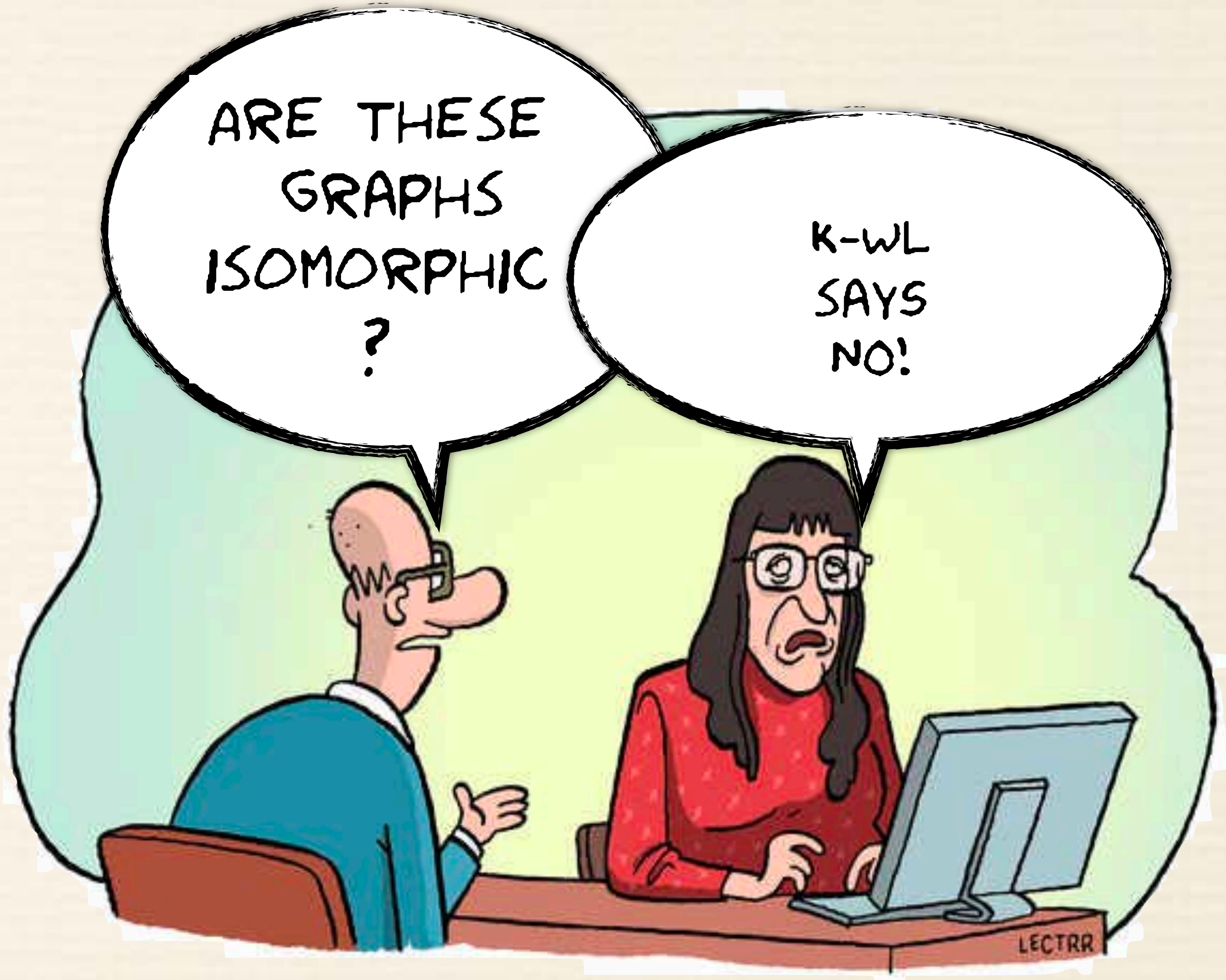
Colour refinement → No → $G \ncong H$

↓

1-WL → No → $G \ncong H$

↓

2-WL → No → $G \ncong H$



More powerful heuristic

Apply heuristic on G and H: If Heuristic say "no" then $G \ncong H$, otherwise we do not know.

$G \cong H$?

Colour refinement → No → $G \ncong H$

↓

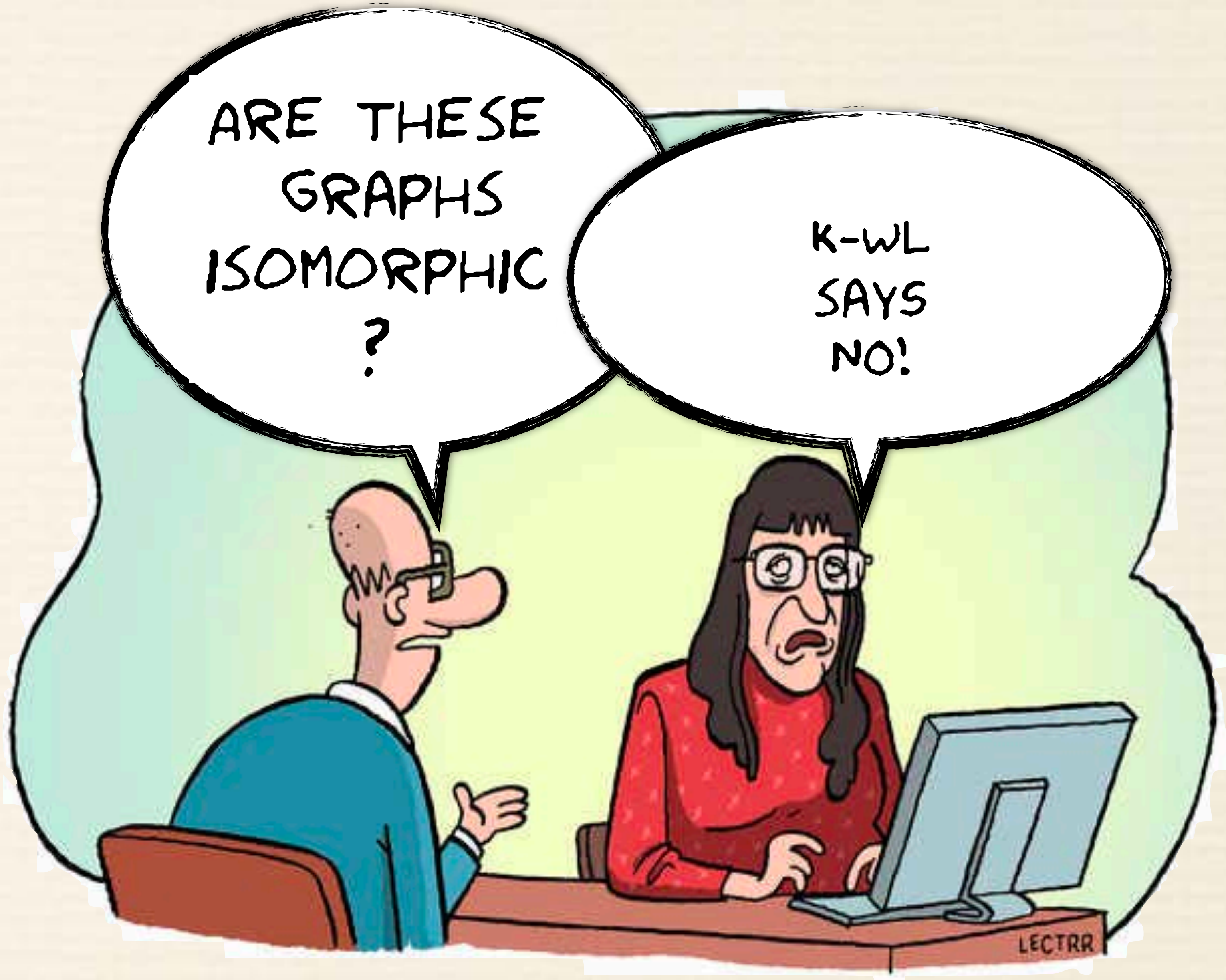
1-WL → No → $G \ncong H$

↓

2-WL → No → $G \ncong H$

↓

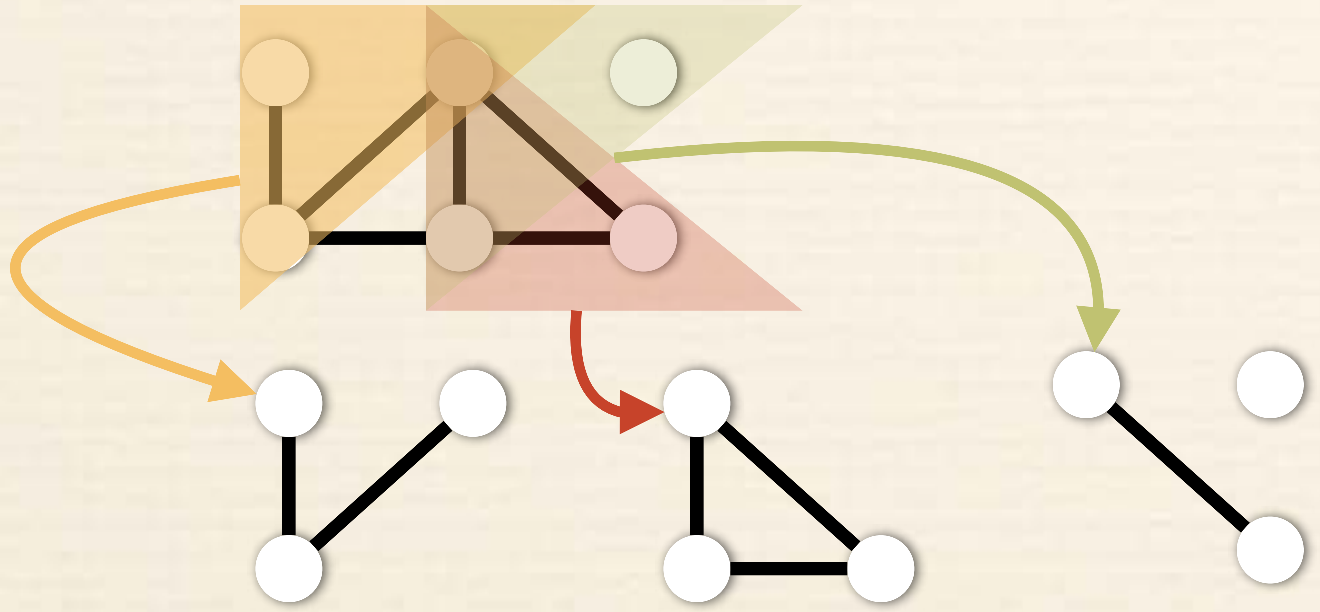
$\vdots$



K-dimensional Weisfeiler-Leman

- Initial: Colour k-tuples of vertices according to label, adjacency and equality information.

  $\Rightarrow$ Same colour if same induced subgraph



Neighbours: two k-tuples $\mathbf{v} = (v_1, \ldots, v_k)$ and $\mathbf{w} = (w_1, \ldots, w_k)$ are i-neighbours if $v_j = w_j$ for all $j \neq i$

K-dimensional Weisfeiler-Leman

- Iteration: k-tuple colour depending on colours of i-neighbours.

$$\mathsf{wl}_k^{(t+1)}(G, v_1, \ldots, v_k) := \Big(\mathsf{wl}_k^{(t)}(G, v_1, \ldots, v_k), M^{(t)}(G, v_1, \ldots, v_k)\Big)$$

$$\begin{aligned} M^{(t)}(G, v_1, \ldots, v_k) := \Big(\, & \mathsf{wl}_{k+1}^{(0)}(v_1, \ldots, v_k, w), \\ & \mathsf{wl}_k^{(t)}(w, v_2, \ldots, v_k), \\ & \vdots \\ & \mathsf{wl}_k^{(t)}(v_1, \ldots, v_{k-1}, w) \mid w \in V_G \Big) \end{aligned}$$

- Graphs: Histogram of colours $\mathsf{wl}_k^{(L)}(G, v, \ldots, v)$ for all $v \in V_G$

Properties of k-WL

Theorem (Cai et al. (1992)

$$\rho(\infty\text{-WL}) \subsetneq \cdots \subsetneq \rho(k\text{WL}) \subsetneq \rho((k-1)\text{-WL}) \subsetneq \cdots \subsetneq \rho(1\text{-WL})$$

Strict increase in power!

Theorem (Cai et al. (1992)

Distinguishability of graphs by k-WL corresponds to distinguishability by (k+1)-variable fragment of FO with counting quantifier (C_{k+1})

- Graphs of size n: Isomorphism problem solved by n-WL
- Large neighbourhoods (nk) and n^k tuples :-(

Cai et al.: *An optimal lower bound on the number of variables for graph identifications.* (1992)
M. Grohe. *The logic of graph neural networks* (2021)

Characterisations of $\rho(k\text{-WL})$

Recall: Theorem (Dell et al. 2018, …)

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all *trees* T

if and only if

colour refinement cannot tell apart G from H

Z. Dvorák: *On recognizing graphs by numbers of homomorphisms* (2010)
Dell et al. *Lovász meets Weisfeiler and Leman* (2018)

Characterisations of $\rho(k\text{-WL})$

Recall: **Theorem** (Dell et al. 2018, …)

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all *trees* T

if and only if

colour refinement cannot tell apart G from H

Now: **Theorem** (Dell et al. 2018, …)

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all graphs T *of tree width k*

if and only if

k-WL cannot tell apart G from H

Z. Dvorák: *On recognizing graphs by numbers of homomorphisms* (2010)

Dell et al. *Lovász meets Weisfeiler and Leman* (2018)

Characterisations of $\rho(k\text{-WL})$

Recall: **Theorem** (Dell et al. 2018, …)

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all *trees* T

if and only if

colour refinement cannot tell apart G from H

Now: **Theorem** (Dell et al. 2018, …)

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all graphs T *of tree width* k

if and only if

k-WL cannot tell apart G from H

Measures "how far from being a tree"

Z. Dvorák: *On recognizing graphs by numbers of homomorphisms* (2010)
Dell et al. *Lovász meets Weisfeiler and Leman* (2018)

Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

k=2

Treewidth

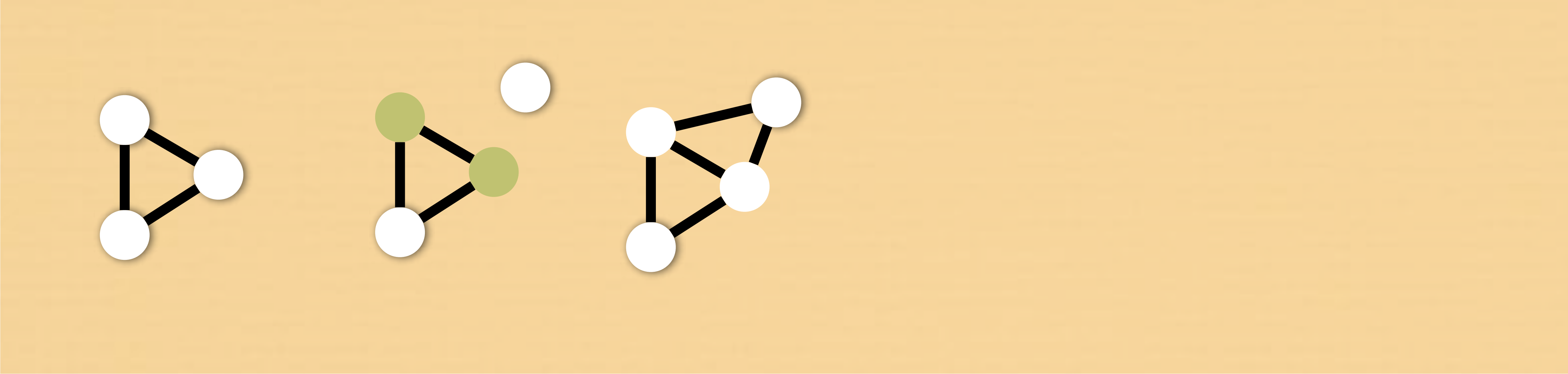
- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

k=2

Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

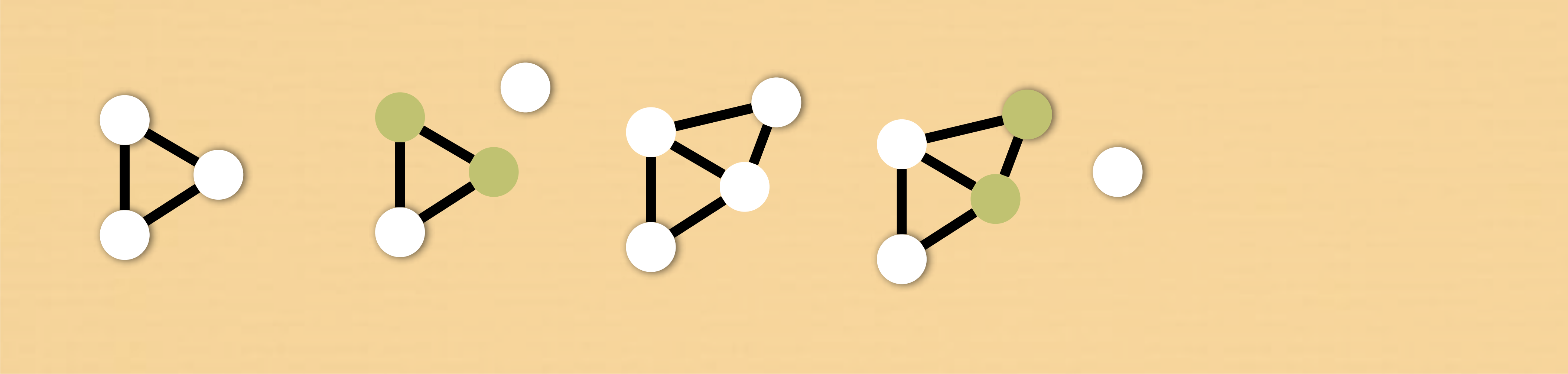
k=2



Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

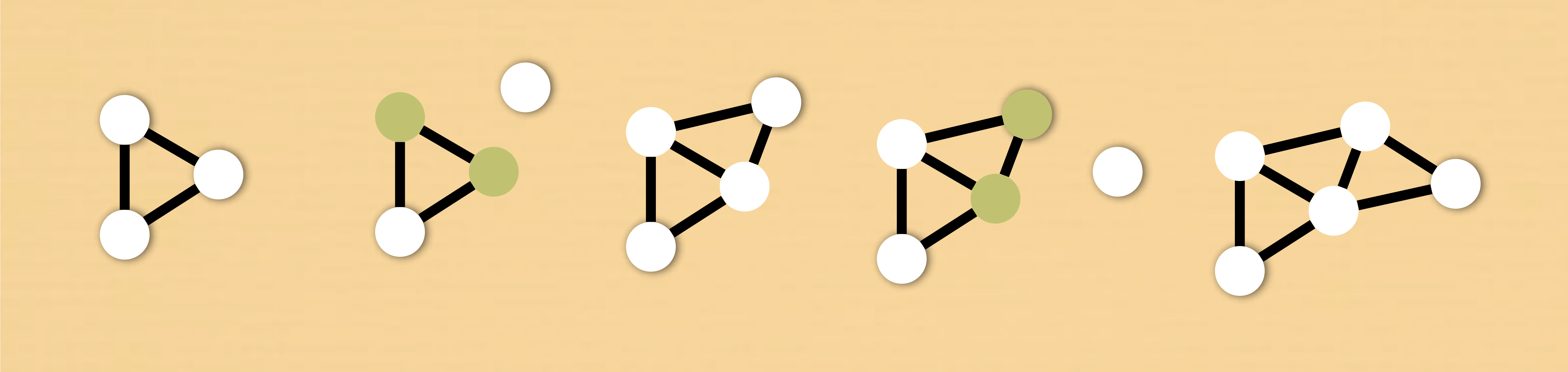
k=2



Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

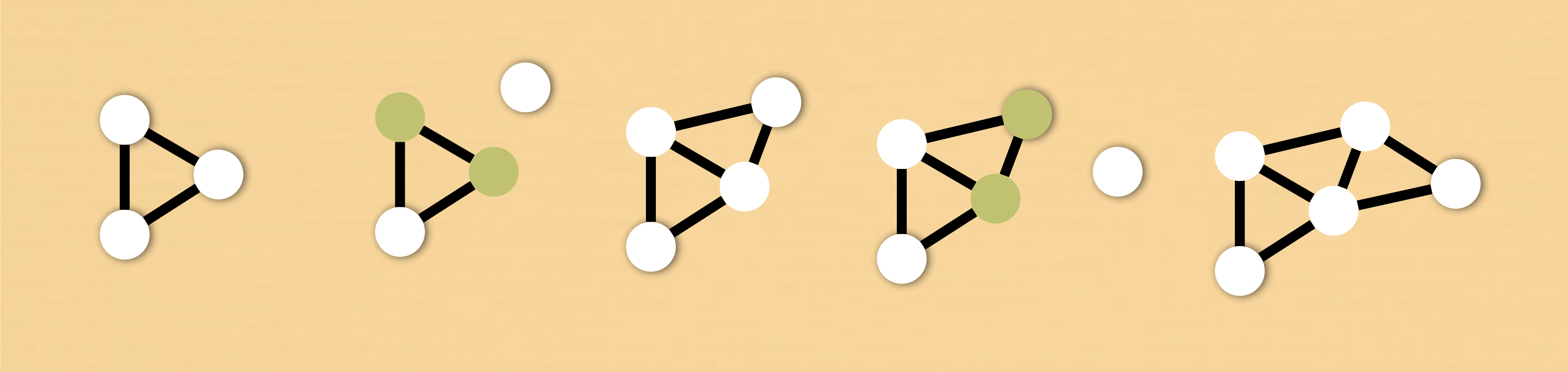
k=2



Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

k=2

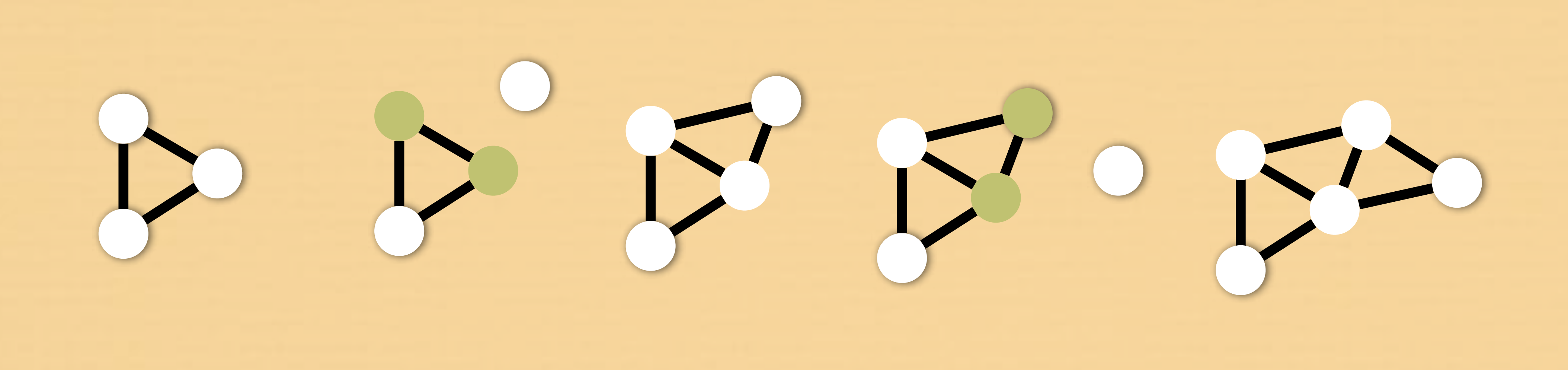


- A partial k-tree is a subgraph of a k-tree

Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

k=2



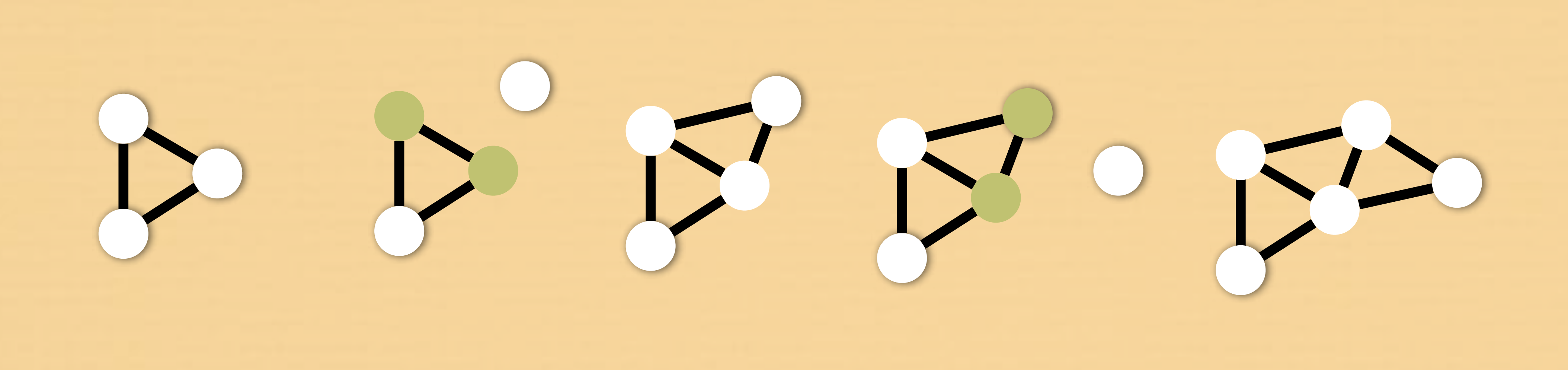
- A partial k-tree is a subgraph of a k-tree

Treewidth of a graph is smallest k such that the graph is a partial k-tree

Treewidth

- A k-tree is a graph that can be obtained starting from a (k+1)-clique and then iteratively adding a vertex connected to a k-clique

k=2

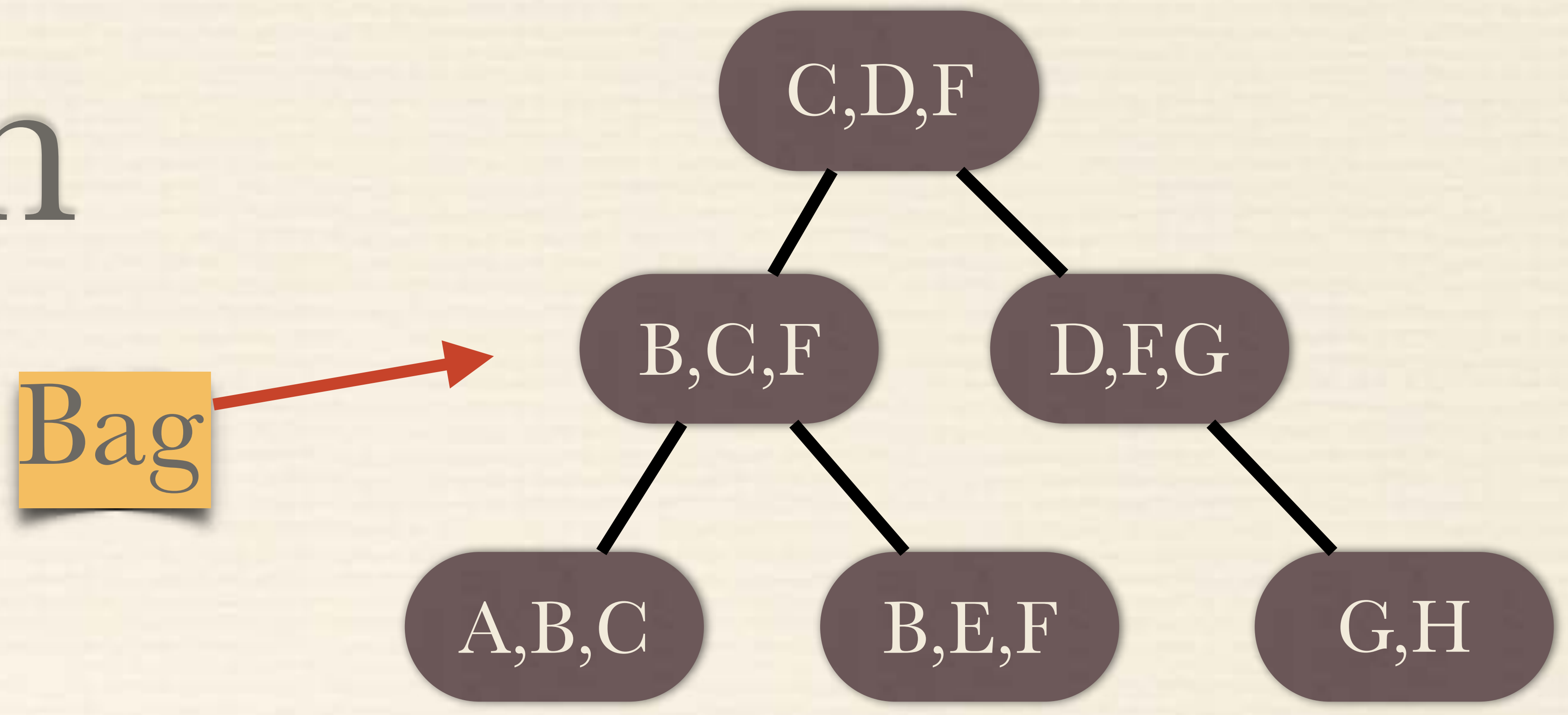


- A partial k-tree is a subgraph of a k-tree

Treewidth of a graph is smallest k such that the graph is a partial k-tree

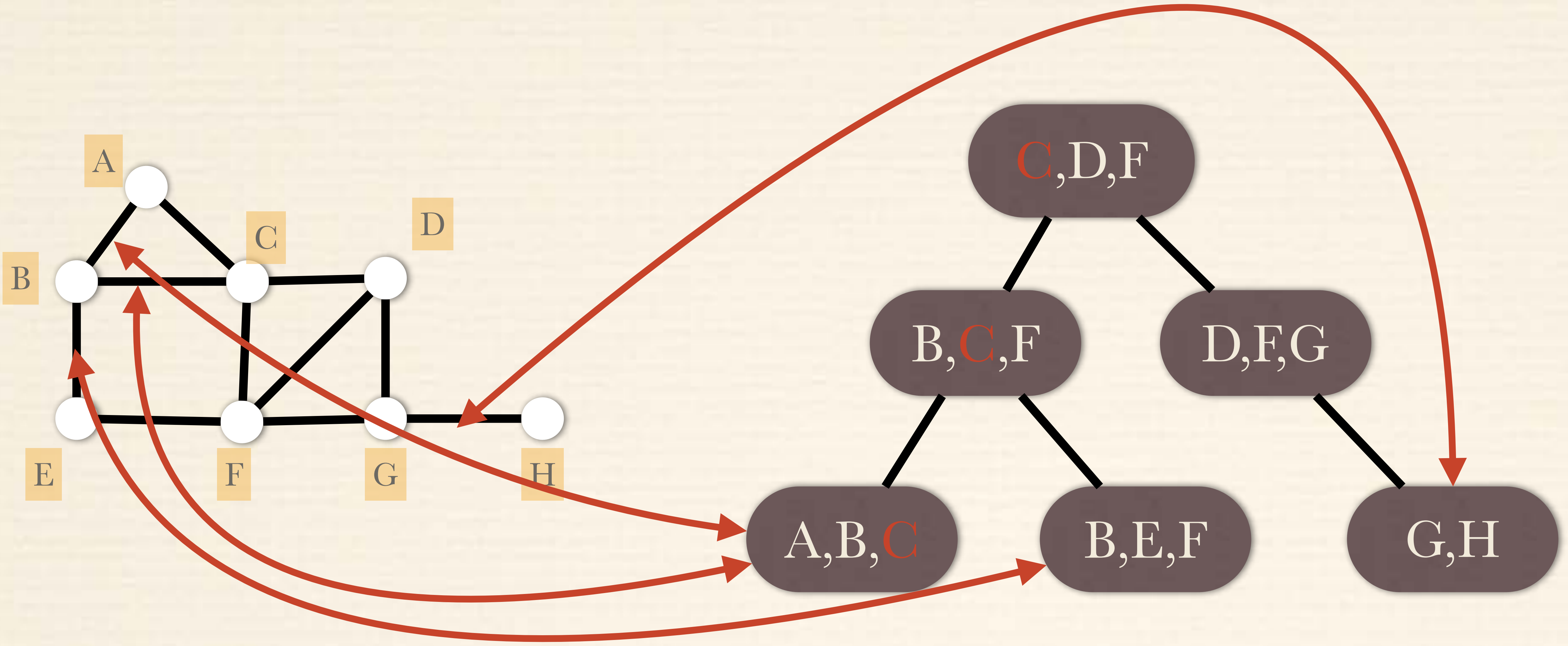
- Trees=Treewidth 1

Treewidth

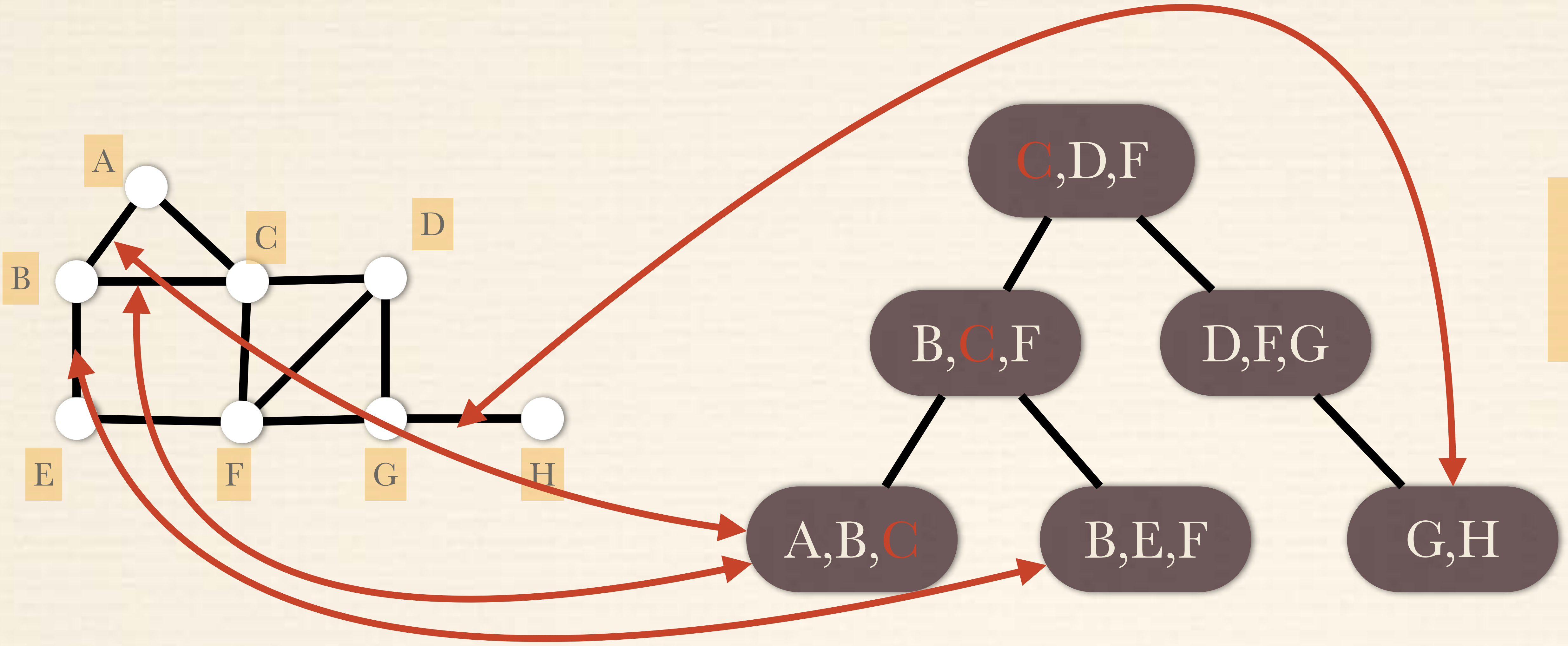


- Alternative definition in terms of tree decomposition
  - If u and v neighbours then there is "bag" containing them both.
  - All bags containing a vertex v from a connected subtree.
  - Graph has treewidth k if it has a tree decomposition with bags of size k+1.

Treewidth

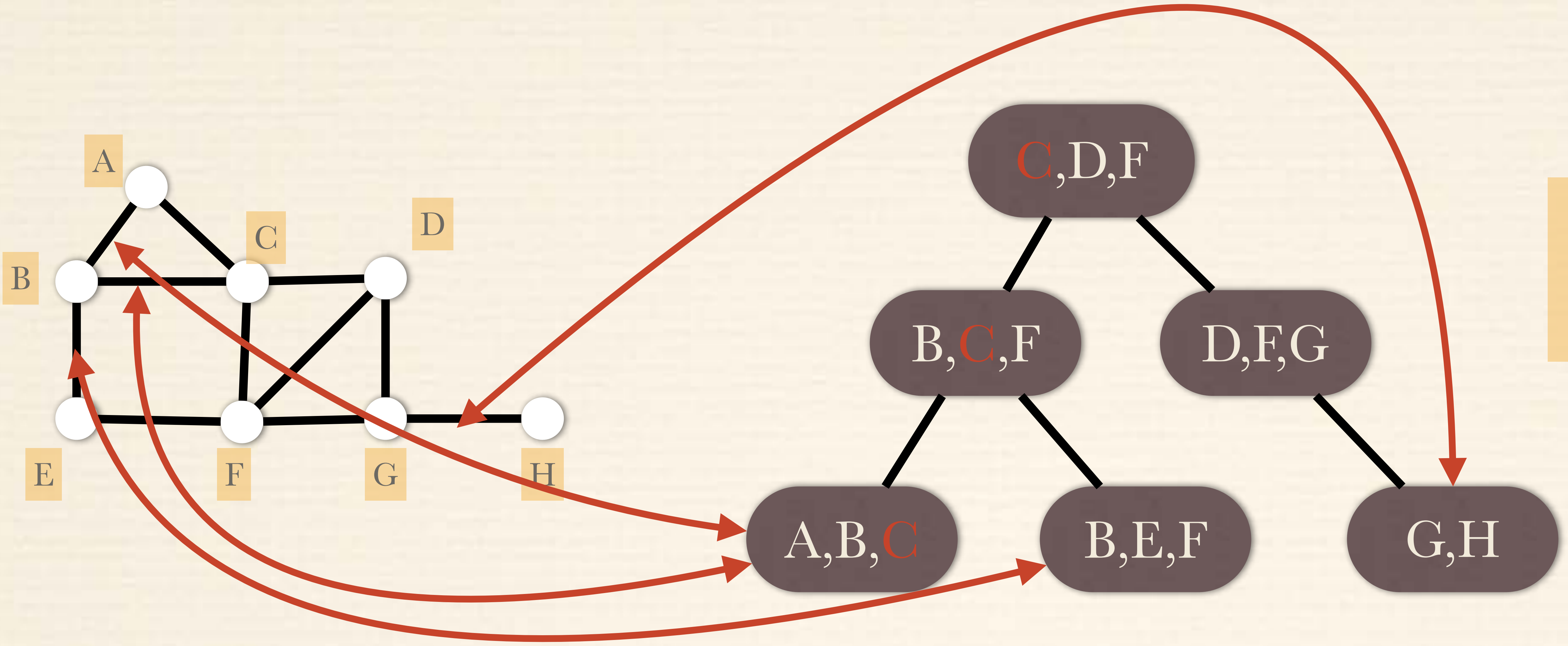


Treewidth

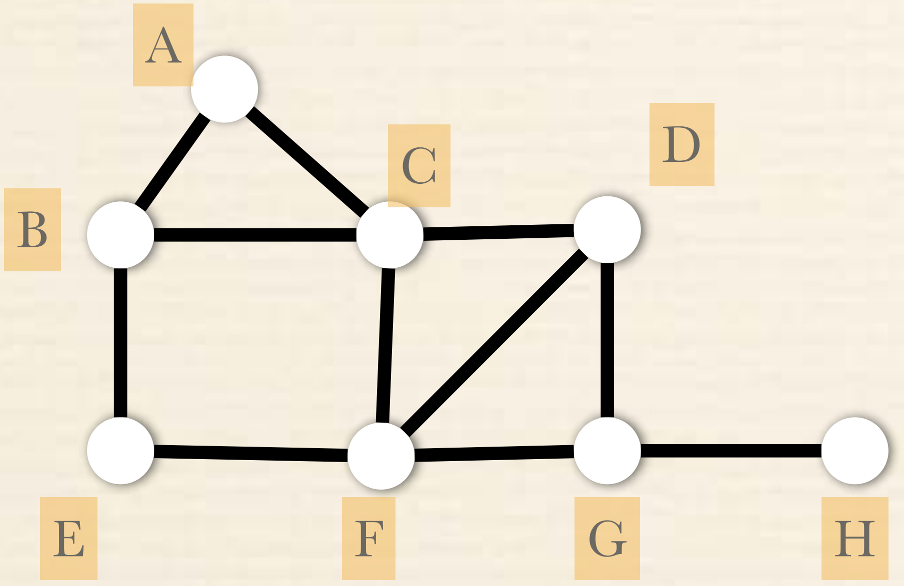


Bag size 3 → Treewidth 2

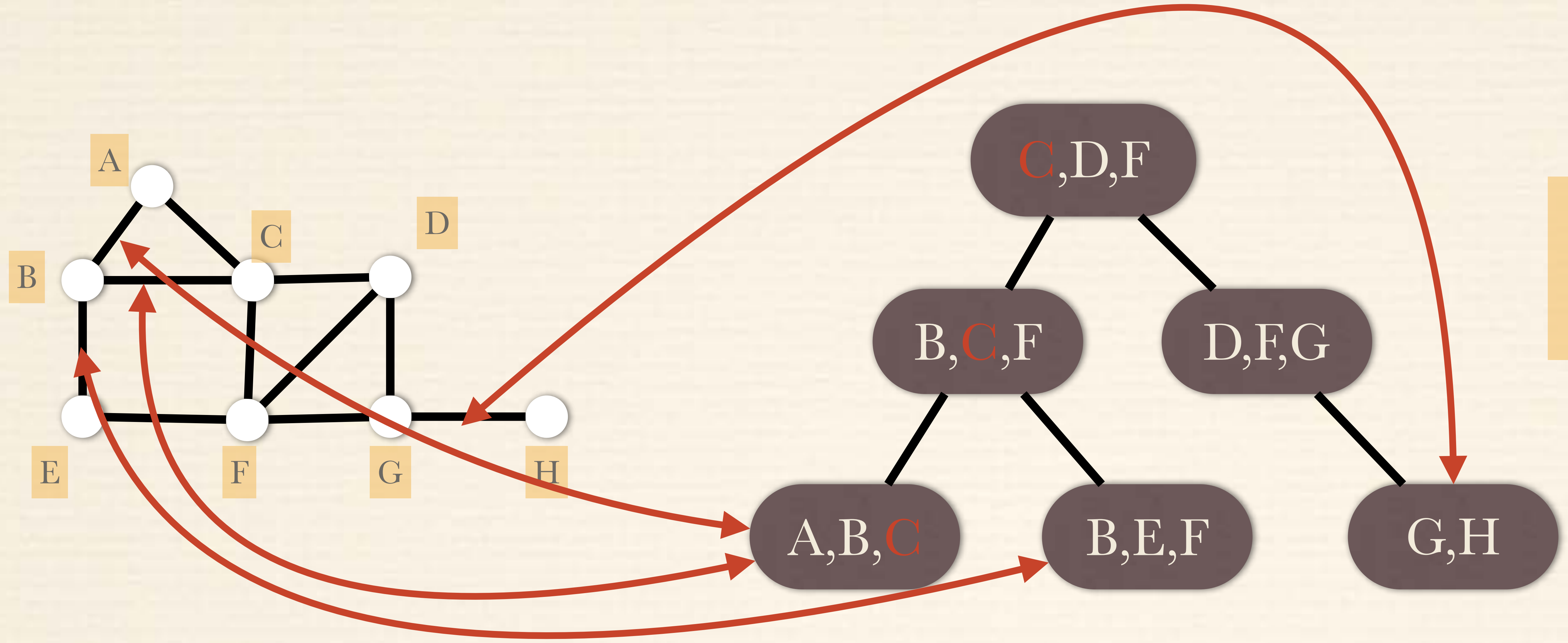
Treewidth



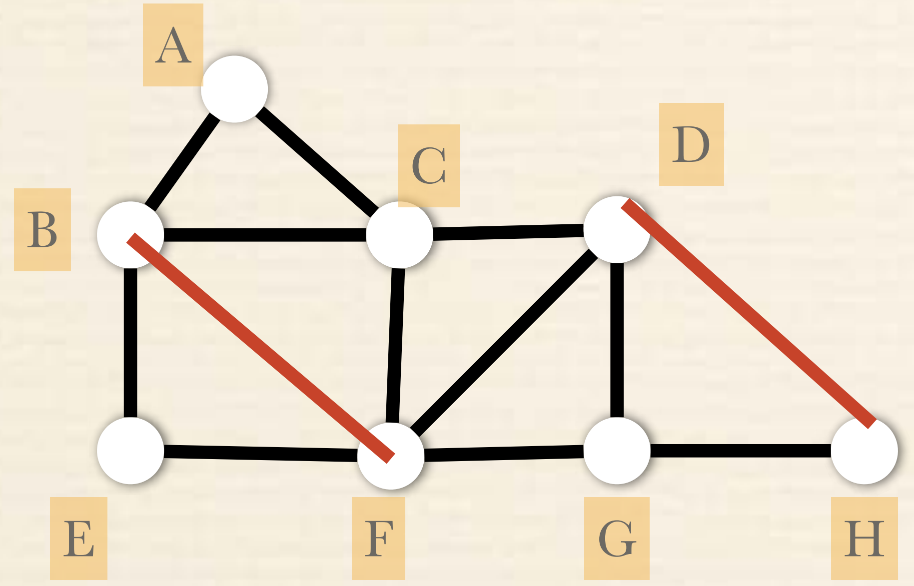
Bag size 3 → Treewidth 2



Treewidth

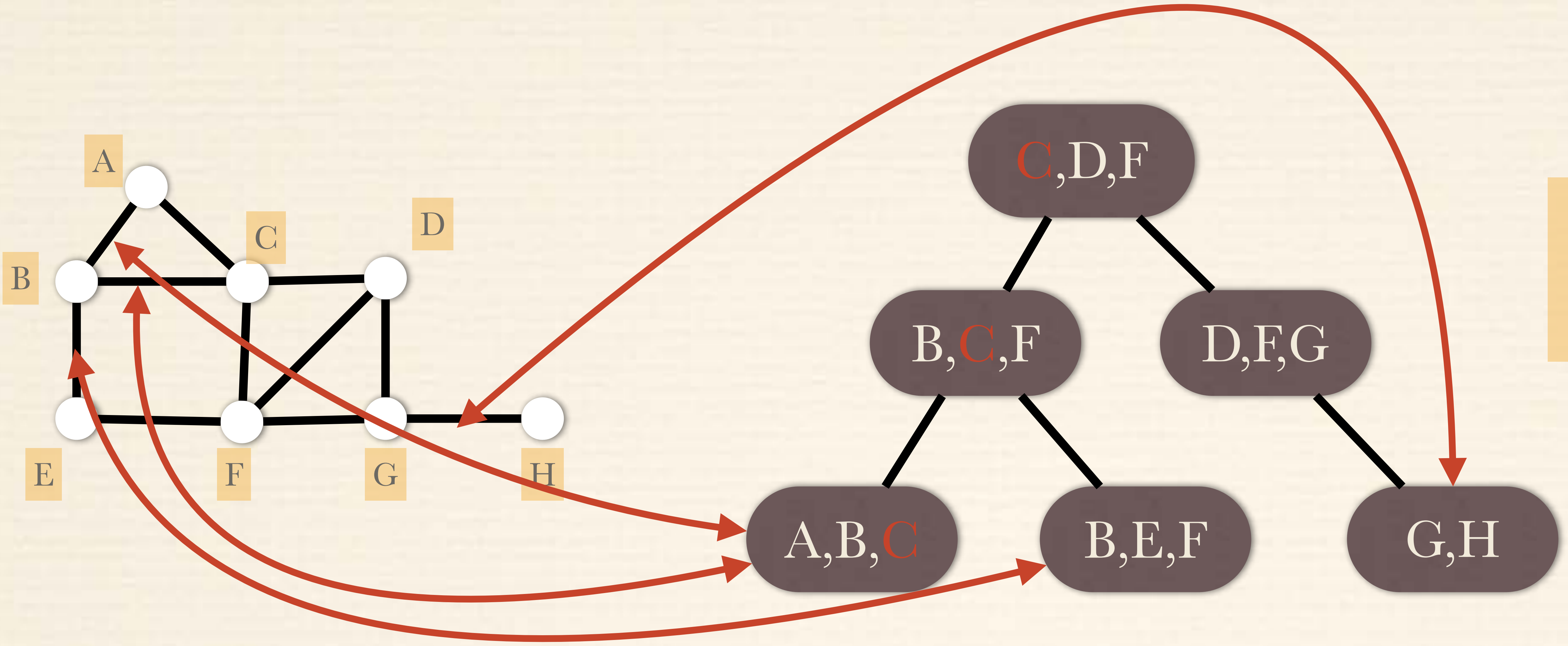


Bag size 3 → Treewidth 2

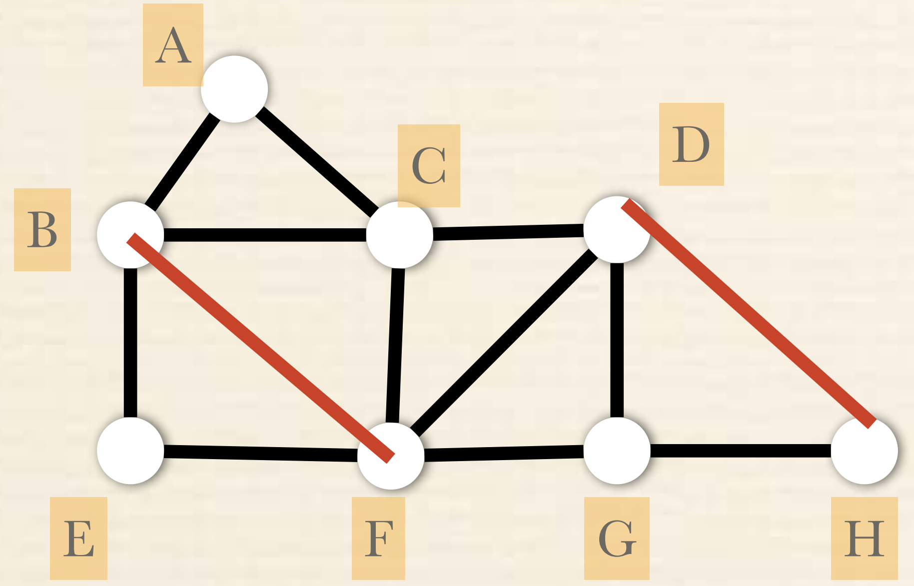


3-tree

Treewidth



Bag size 3 → Treewidth 2



3-tree

Question:

- tw(cycle of length k)?
- tw(k-clique)?

Back to $\mathscr{P}$-MPNNs and $\mathscr{P}$-GSNs

Back to $\mathscr{P}$-MPNNs and $\mathscr{P}$-GSNs

$\mathscr{P}$-MPNNs

$$\xi^{(0)}(G,v) := \text{Hot-one encoding of label of vertex } v + \mathsf{hom}(P^r, G^v), \ldots, \mathsf{hom}(P^r_\ell, G^v)$$

$$\xi^{(t)}(G,v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G,v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G,v), \xi^{(t)}(G,u), \mathsf{hom}(P^r, G^u), \ldots, \mathsf{hom}(P^r_\ell, G^u) \mid u \in N_G(v)\}\}\big)\Big)$$

$$\rho(G) := \mathsf{Readout}\Big(\{\{\xi^{(L)}(G,v) \mid v \in V_G\}\}\Big)$$

Back to $\mathscr{P}$-MPNNs and $\mathscr{P}$-GSNs

$\mathscr{P}$-MPNNs

$$\xi^{(0)}(G,v) := \text{Hot-one encoding of label of vertex } v + \mathsf{hom}(P^r, G^v), \ldots, \mathsf{hom}(P^r_\ell, G^v)$$

$$\xi^{(t)}(G,v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G,v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G,v), \xi^{(t)}(G,u), \mathsf{hom}(P^r, G^u), \ldots, \mathsf{hom}(P^r_\ell, G^u) \mid u \in N_G(v)\}\}\big)\Big)$$

$$\rho(G) := \mathsf{Readout}\Big(\{\{\xi^{(L)}(G,v) \mid v \in V_G\}\}\Big)$$

Theorem

$\mathsf{hom}(T,G) = \mathsf{hom}(T,H)$ for all $\mathscr{P}$–*pattern trees* T, *if and only* no $\mathscr{P}$-MPNN can distinguish G from H.

Back to $\mathscr{P}$-MPNNs and $\mathscr{P}$-GSNs

$\mathscr{P}$-MPNNs

$\xi^{(0)}(G, v) :=$ Hot-one encoding of label of vertex v + $\mathsf{hom}(P^r, G^v), \ldots, \mathsf{hom}(P^r_\ell, G^v)$

$$\xi^{(t)}(G, v) := \mathsf{Upd}^{(t)}\Big(\xi^{(t-1)}(G, v), \mathsf{Agg}^{(t)}\big(\{\{\xi^{(t-1)}(G, v), \xi^{(t)}(G, u), \mathsf{hom}(P^r, G^u), \ldots, \mathsf{hom}(P^r_\ell, G^u) \mid u \in N_G(v)\}\}\big)\Big)$$

$$\rho(G) := \mathsf{Readout}\Big(\{\{\xi^{(L)}(G, v) \mid v \in V_G\}\}\Big)$$

Theorem

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all $\mathscr{P}$*–pattern trees* T*, if and only* no $\mathscr{P}$-MPNN can distinguish G from H.

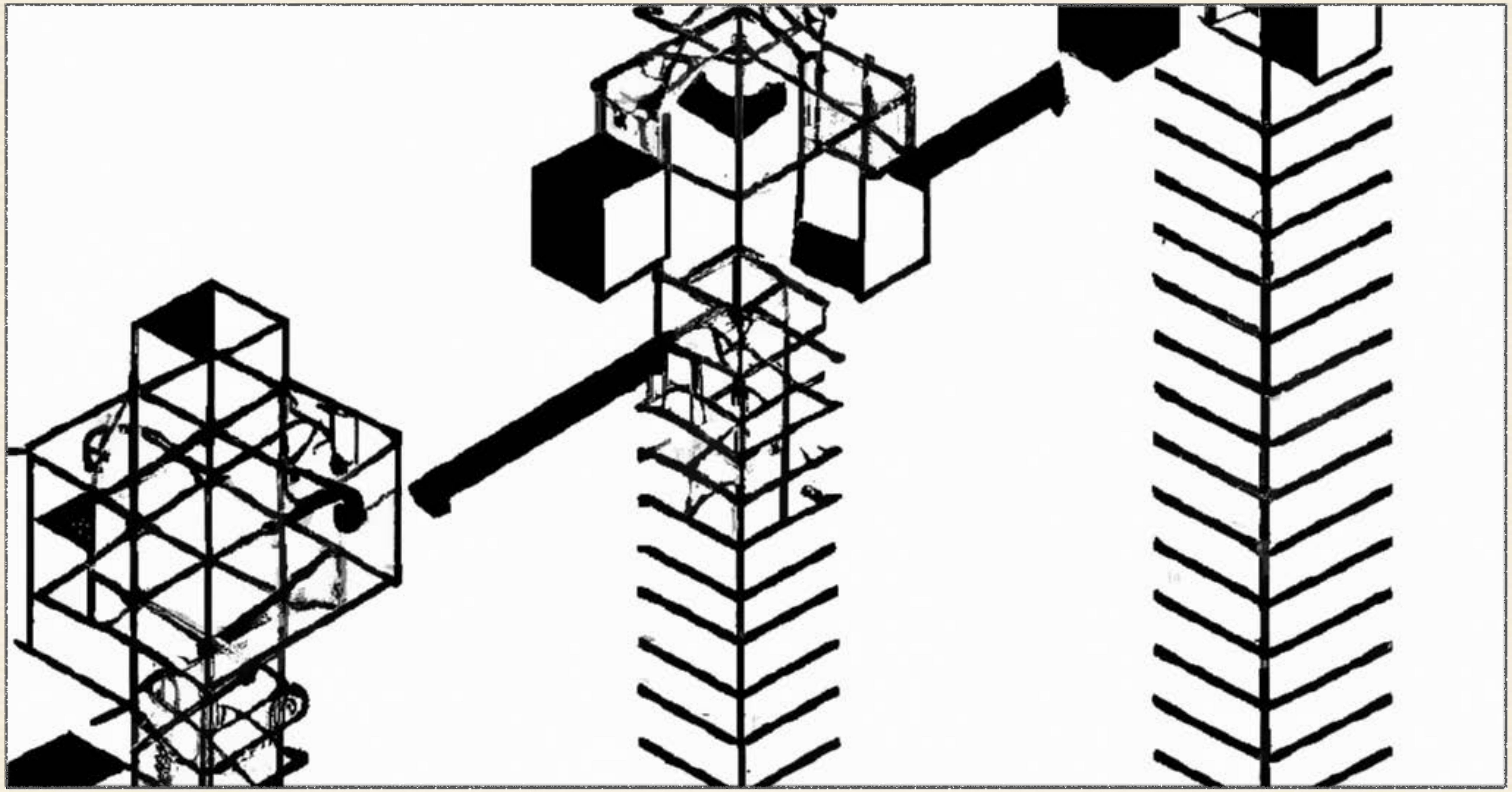
Theorem

If the patterns P in $\mathscr{P}$ have maximal tree width k then the power of $\mathscr{P}$-MPPNs is bounded by k-WL. Similar result for $\mathscr{P}$-GSN using $\mathscr{P}^\star$-MPNNs.

$\mathscr{P}_k$ max tree width k

$\mathscr{P}^\star_k$ max tree width k

1-rMPNN+
randomness clouds
k-rMPNN+
∞ − rMPNN+
$\mathscr{P}^{\star}_{\infty}$-SGNs
$\mathscr{P}_{\infty}$-MPNNs
k-WL
$\mathscr{P}^{\star}_{k}$-SGNs
(K-1)-OSAN
$\mathscr{P}_{k}$-MPNNs
2-WL
$\mathscr{P}^{\star}_{2}$-SGNs
$\mathscr{P}_{2}$-MPNNs
DSS-GNN
DS-GNN/1-OSAN/1-OSAN^t
(1,1)-WL
SpMPNNs
MPNN+
SpMPNNs(inv)
GNN-AK
1-WL
Colour Refinement
NGNN
Reconstruction GNN
GCN
MPNNs
ID-GNN
$\mathscr{H}$
GIN
DropoutGNN
Message passing
Expressiveness
Complexity



Higher-order GNNs

Neural implementations of k-WL

Idea: higher-order GNNs

Theorem (Dell et al. 2018, …)

$\mathrm{hom}(T, G) = \mathrm{hom}(T, H)$ for all graphs T *of tree width* k
if and only if
k-WL cannot tell apart G from H

1-WL → MPNNs

Z. Dvorák: *On recognizing graphs by numbers of homomorphisms* (2010)
Dell et al. *Lovász meets Weisfeiler and Leman* (2018)

Idea: higher-order GNNs

Theorem (Dell et al. 2018, …)

$\mathsf{hom}(T, G) = \mathsf{hom}(T, H)$ for all graphs T *of tree width* k

if and only if

k-WL cannot tell apart G from H

$$\begin{tikzcd} \text{1-WL} \arrow[r] \arrow[d] & \text{MPNNs} \arrow[d] \\ \text{k-WL} \arrow[r] & \text{k-MPNNs} \end{tikzcd}$$

Z. Dvorák: *On recognizing graphs by numbers of homomorphisms* (2010)
Dell et al. *Lovász meets Weisfeiler and Leman* (2018)

Idea: higher-order GNNs

Theorem (Dell et al. 2018, …)

$\hom(T, G) = \hom(T, H)$ for all graphs T *of tree width* k

if and only if

k-WL cannot tell apart G from H

$$\begin{tikzcd} \text{1-WL} \arrow[r] \arrow[d] & \text{MPNNs} \arrow[d] \\ \text{k-WL} \arrow[r] & \text{k-MPNNs} \end{tikzcd}$$

k-MPNNs will detect more graph information than MPNNs

Z. Dvorák: *On recognizing graphs by numbers of homomorphisms* (2010)
Dell et al. *Lovász meets Weisfeiler and Leman* (2018)

k-Folklore GNNs (k-FGNs)

$$\xi^{(t)}(G, v_1, \ldots, v_k) := \mathsf{MLP}_1^{(t)}\Big(\sum_{u \in V_G} \prod_{i=1}^{k} \mathsf{MLP}_2^{(t)}(\xi^{(t-1)}(G, v_1, \ldots, v_{i-1}, u, v_{i+1}, \ldots, v_k))\Big)$$

k-vertex embedding   Global aggregation   Uses multiplication

Expressive power?

Maron et al.: *Provably powerful graph networks* (2019)
W. Azizian and M. Lelarge. *Characterizing the expressive power of invariant and equivariant graph neural networks* (2021)

k-Folklore GNNs (k-FGNs)

$$\xi^{(t)}(G, v_1, \ldots, v_k) := \mathsf{MLP}_1^{(t)}\Big(\sum_{u \in V_G} \prod_{i=1}^{k} \mathsf{MLP}_2^{(t)}(\xi^{(t-1)}(G, v_1, \ldots, v_{i-1}, u, v_{i+1}, \ldots, v_k))\Big)$$

k-vertex embedding   Global aggregation   Uses multiplication

Expressive power?

Theorem (Maron et al. 2019), Azizian and Lelarge 2021)

$$\rho(k - \text{FGNN}) = \rho(k\text{-WL})$$

Maron et al.: *Provably powerful graph networks* (2019)
W. Azizian and M. Lelarge. *Characterizing the expressive power of invariant and equivariant graph neural networks* (2021)

k-GNNs

A simpler architecture:

$$\xi^{(t)}(G, v_1, \ldots, v_k) := \sigma\Bigg(\xi^{(t-1)}(G, v_1, \ldots, v_k)\mathbf{W}_1^{(t)} + \Big(\sum_{i=1}^{k} \boxed{\sum_{u \in V_G}} \xi^{(t)}(G, v_1, \ldots, v_{i-1}, u, v_{i+1}, \ldots, v_k)\Big)\mathbf{W}_2^{(t)} \Bigg)$$

Global aggregation

Expressive power?

Morris et al.: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019)

k-GNNs

A simpler architecture:

$$\xi^{(t)}(G, v_1, \ldots, v_k) := \sigma\left(\xi^{(t-1)}(G, v_1, \ldots, v_k)\mathbf{W}_1^{(t)} + \Big(\sum_{i=1}^{k} \boxed{\sum_{u \in V_G}} \xi^{(t)}(G, v_1, \ldots, v_{i-1}, u, v_{i+1}, \ldots, v_k)\Big)\mathbf{W}_2^{(t)} \right)$$

Global aggregation

Expressive power?

Theorem (Morris et al. 2019)

$$\rho(k - \mathrm{GNN}) = \rho(k\text{-WL})$$

Morris et al.: *Weisfeiler and Leman go neural: Higher-order graph neural networks.* (2019)

Linear equivariant layers

$L : \mathbb{R}^{n^k} \rightarrow \mathbb{R}^{n^\ell}$ s.t. $L(\mathbf{P}^t\mathbf{X}\mathbf{P}) = \mathbf{P}^t L(\mathbf{X})\mathbf{P}$ for all permutation matrices $\mathbf{P}$

One can find a basis $\mathbf{B}_\gamma$ s.t. $\mathbf{L} = \sum_\gamma a_\gamma \mathbf{B}_\gamma$

$n = 16, k = \ell = 2$

➞ Build higher-order GNN using linear equivariant layers

Maron et al: *Invariant and equivariant graph networks* (2019)

k-IGNs

$$\xi^{(t)}(G, v_1, \ldots, v_k) := \sigma\Big(\sum_{\gamma} \sum_{w_1,\ldots,w_k} \mathbf{B}_{\gamma} \mathbf{W}_{\gamma}^{(t)} \xi^{(t-1)}(G, w_1, \ldots, w_k) + \sum_{\mu} \mathbf{B}_{\mu} \mathbf{W}_{\mu}^{(t)} \Big)$$

Equality types ~ linear equivariant basis

Theorem (Maron et al. 2019, G. and Reutter 2022)

$$\rho(k - \text{IGN}) = \rho((k-1)\text{-WL})$$

Maron et al.: *Provably powerful graph networks* (2019)
G. and J. Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)

1-rMPNN+
randomness clouds
k-rMPNN+
∞ − rMPNN+
$\mathscr{P}^\star_\infty$-SGNs
$\mathscr{P}_\infty$-MPNNs
(k+1)-GNNs
k-IGNs
k-WL
k-GNNs
$\mathscr{P}^\star_k$-SGNs
(K-1)-OSAN
k-FGNNs
$\mathscr{P}_k$-MPNNs
(K-1)IGNs
$\mathscr{P}^\star_2$-SGNs
2-IGNs
2-WL
2-GNNs
DSS-GNN
$\mathscr{P}_2$-MPNNs
DS-GNN/1-OSAN/1-OSAN^t
(1,1)-WL
SpMPNNs
MPNN+
SpMPNNs(inv)
GNN-AK
1-WL
Colour Refinement
NGNN
Reconstruction GNN
GCN
MPNNs
ID-GNN
$\mathscr{H}$
GIN
DropoutGNN
Message passing
Expressiveness
Complexity

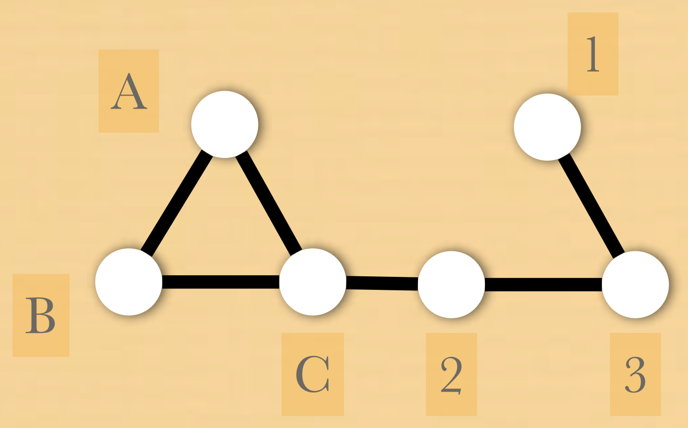
Higher-order methods

- Do not scale well, but are expressive
- Do not leverage sparsity of graphs
- Powerful, but leads to overfitting

There are several attempts to make them scalable without sacrificing power.

Morris et al.: *Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings (2020)*
Morris et al.: *SpeqNets: Sparsity-aware Permutation-equivariant Graph Networks.* (2022)

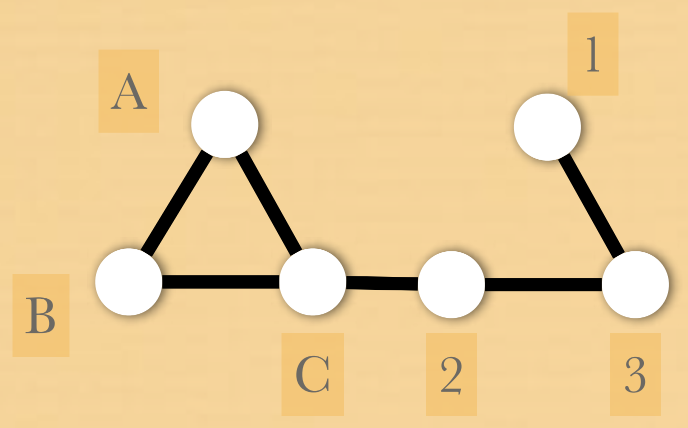
"Local" k-GNNs: k-LGNNs



(A,A)
(A,B)
(A,C)
(A,1)
(A,2)
Neighbours (A,3)
(A,B) → (A,B)
(B,B)
(C,B)
(1,B)
(2,B)
(3,B)

Morris et al.: *Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings (2020)*

“Local” k-GNNs: k-LGNNs



Only when edge

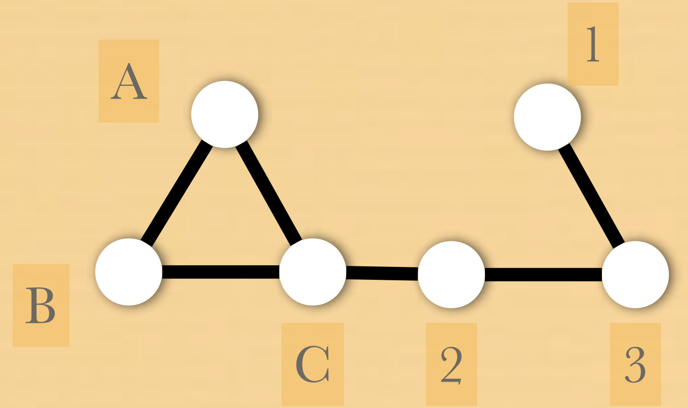
(A,A)
(A,B)
(A,C)
(A,1)
(A,2)
(A,3)

Neighbours

(A,B) → (A,B)
(B,B)
(C,B)
(1,B)
(2,B)
(3,B)

Morris et al.: *Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings (2020)*

"Local" k-GNNs: k-LGNNs



Neighbours

(A,B) → (A,A), (A,B), (A,C), (A,1), (A,2), (A,3), (A,B), (B,B), (C,B), (1,B), (2,B), (3,B)

Only when edge

With B

With A

→ (A,A), (A,B), (A,C), ~~(A,1)~~, ~~(A,2)~~, ~~(A,3)~~, (A,B), (B,B), (C,B), ~~(1,B)~~, ~~(2,B)~~, ~~(3,B)~~

Morris et al.: *Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings (2020)*

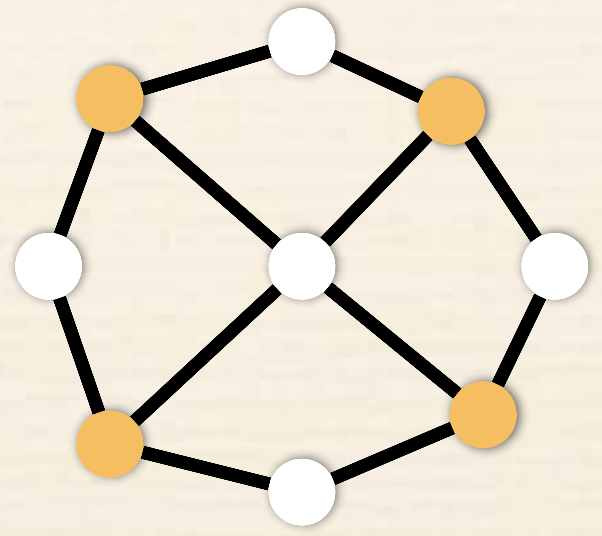
k-LGNNs

$$\xi^{(t)}(G, v_1, \ldots, v_k) := \sigma\left(\xi^{(t-1)}(G, v_1, \ldots, v_k)\mathbf{W}_1^{(t)} + \Big(\sum_{i=1}^{k} \sum_{(u,v_i)\in E_G} \xi^{(t)}(G, v_1, \ldots, v_{i-1}, u, v_{i+1}, \ldots, v_k)\Big)\mathbf{W}_2^{(t)} \right)$$

Theorem (Morris et al. (2020), G and Reutter (2022)

$$\rho((k+1)\text{-WL}) \subsetneq \rho(k - \text{LGNN}) \subsetneq \rho(k\text{-WL})$$

Can detect distance two (k+1)-cliques



Morris et al.: *Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings (2020)*
G. and J. Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)


1-rMPNN+
randomness clouds
k-rMPNN+
∞ − rMPNN+
$\mathcal{P}^{\star}_{\infty}$-SGNs
$\mathcal{P}_{\infty}$-MPNNs
k-LGNNs
k-WL
k-IGNs
(k+1)-GNNs
k-GNNs
$\mathcal{P}^{\star}_{k}$-SGNs
(K-1)-OSAN
k-FGNNs
$\mathcal{P}_{k}$-MPNNs
(K-1)IGNs
2-LGNNs
$\mathcal{P}^{\star}_{2}$-SGNs
2-WL
2-GNNs
DSS-GNN
$\mathcal{P}_{2}$-MPNNs
DS-GNN/1-OSAN/1-OSAN^{t}
SpMPNNs
(1,1)-WL
MPNN+
SpMPNNs(inv)
GNN-AK
1-WL
Colour Refinement
NGNN
GCN
MPNNs
Reconstruction GNN
ID-GNN
$\mathcal{H}$
GIN
Message passing
DropoutGNN
Expressiveness
Complexity

Let’ stop filling in the landscape

Conclusions

And look ahead

Semi-conclusion

- Expressivity has been an important concept in graph learning since 2019
- Has been pushing forward the area: different techniques to boost power:
  - k-WL, feature augmentation, subgraphs, structured modulated message passing, ….
- Expressive models juggle with
  - Complexity, overfitting, …

Semi-conclusion

- When methods are shown to be powerful: existential proofs.
- No reason that this power is met in practice.
- Also, distinguishing power is necessary but not sufficient in practice…

Semi-conclusion

- Expressivity has been an important concept in graph learning since 2019
- Has been pushing forward the area: different techniques to boost power:
  - k-WL, feature augmentation, subgraphs, structured modulated message passing, ….
- Expressive models juggle with
  - Complexity, overfitting, …

What to use?

Subgraph	Feature Augmentation	Higher-order
❖ Small graphs ❖ Good compromise in general	❖ Large training datasets ❖ Invariance not importnat ❖ Preprocessing ok	❖ Graphs are small ❖ Efficiency not essential ❖ Expressivity guarantee needed

Road ahead

Expressiveness

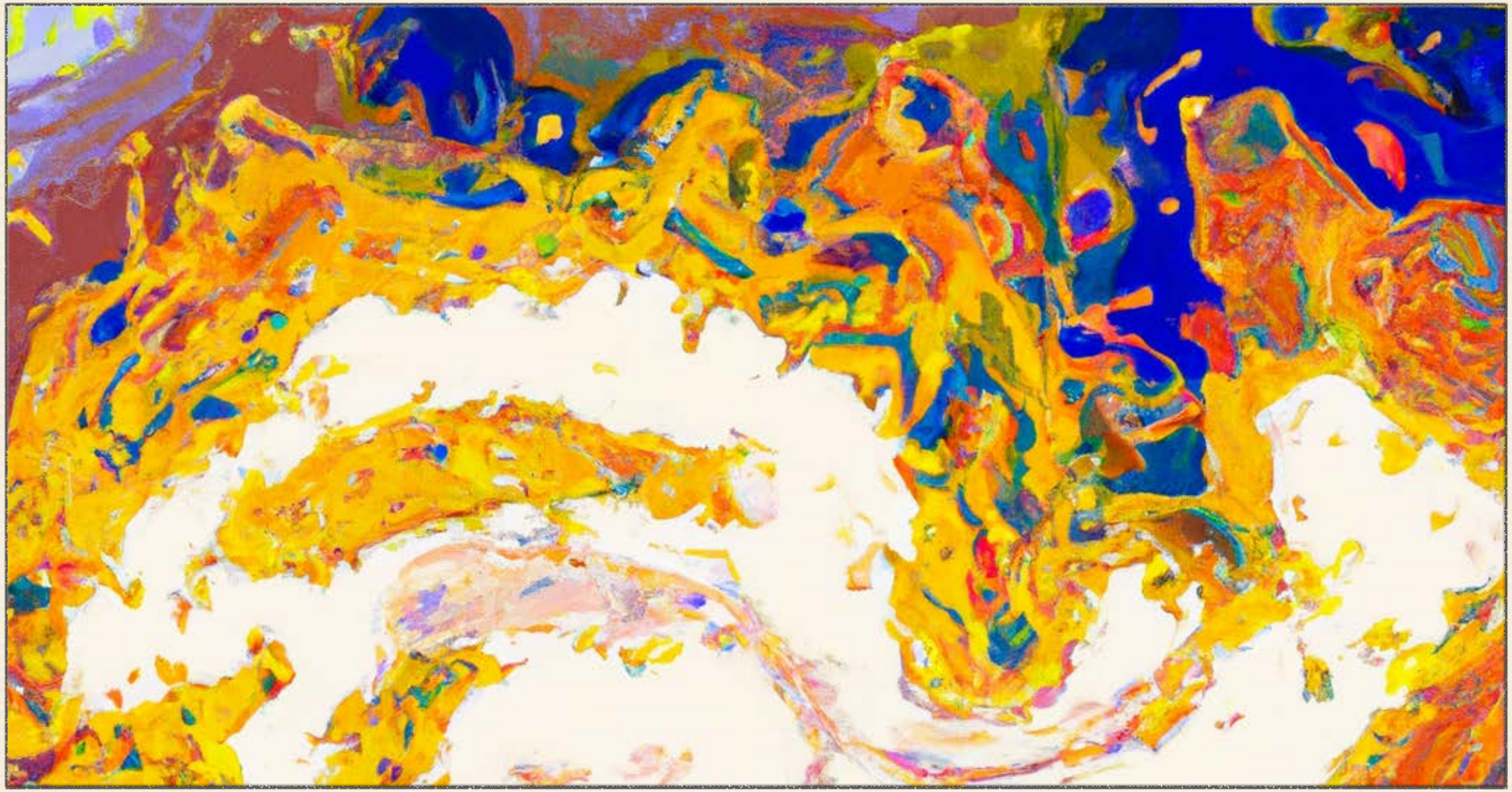
- A lot of recent (2023 progress)
- WL hierarchy needs better reconciliation with practice
- Hom count characterisations
- Relational

Connection with Learning??

- Optimisation and training unexplored

- Generalisation properties
- Sample efficiency?

Gary et al. :*Generalization and Representational Limits of Graph Neural Networks* (2020)
Morris et al: *WL meet VC* (2023)



$x^2/2pi$

Bounding embedding methods

An "easy" way to analyse the power of graph embeddings

How to get k-WL bounds?

Without knowing k-WL?

Higher-order MPNNs

- They are a generalisation of classical MPNNs.
- They provide a flexible mechanism to describe various graph learning architectures.

Easy way to obtain upper bounds on the expressive power of graph learning architectures.

G. and J. Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)

Higher-order MPNNs

- Higher-order MPNNs are defined inductively and declaratively.
- We provide *syntax* and *semantics.*
- With each higher-order MPNN φ we associate:
  - A dimension describing the *output feature dimension*; and
  - A set of free variables and we write $\varphi(\mathbf{x})$ with $\mathbf{x} = \{x_1, \ldots, x_\ell\}$.

Higher-order MPNNs

Syntax

Higher-order MPNN

$\varphi(\mathbf{x})$ of dimension d and free variables $\mathbf{x} = \{x_1, \ldots, x_\ell\}$

Semantics

Higher-order embedding

$$\xi_\varphi : \mathcal{G} \to (\mathcal{V}^\ell \to \mathbb{R}^d) : (G, v_1, \ldots, v_\ell) \mapsto \mathbb{R}^d$$

Higher-order MPNNs: Atomic

Atomic higher-order MPNNs: Syntax

Label: $\varphi(x_i) := \mathsf{Lab}_j(x_i)$ of dim 1 and free var x_i

Edge: $\varphi(x_i, x_j) := E(x_i, x_j)$ of dim 1, free vars x_i, x_j

Equality: $\varphi(x_i, x_j) := \mathbf{1}[x_i = x_j]$ of dim 1, free vars x_i, x_j

Higher-order MPNNs: Atomic

Atomic higher-order MPNNs: Syntax

Label: $\varphi(x_i) := \mathsf{Lab}_j(x_i)$ of dim 1 and free var x_i

Edge: $\varphi(x_i, x_j) := E(x_i, x_j)$ of dim 1, free vars x_i, x_j

Equality: $\varphi(x_i, x_j) := \mathbf{1}[x_i = x_j]$ of dim 1, free vars x_i, x_j

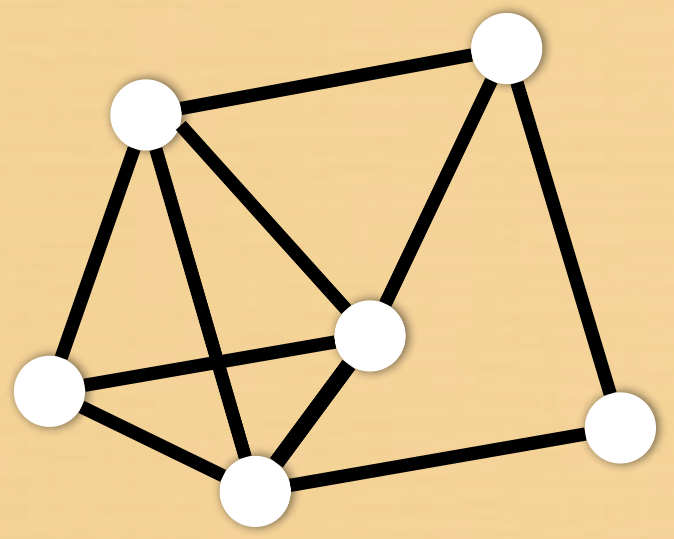
Semantics

$\xi_\varphi : (v_1, v_2, \ldots, v_p) \mapsto j\text{th feature of } v_i$

$$\xi_\varphi : (v_1, v_2, \ldots, v_p) \mapsto \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$\xi_\varphi : (v_1, v_2, \ldots, v_p) \mapsto \begin{cases} 1 & v_i = v_j \\ 0 & \text{otherwise} \end{cases}$$

Higher-order MPNNs: Atomic



$$\varphi(v_1, v_2) := E(x_1, x_2)$$

$$\xi_\varphi : (v_1, v_2) \mapsto \begin{cases} 1 & (v_1, v_2) \text{ is a edge} \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Higher-order MPNNs: Function Application

Function application: Syntax

Let $\varphi_1(\mathbf{x}_1), \ldots, \varphi_\ell(\mathbf{x}_1)$ be higher-order MPNNs of dim $d_1, \ldots, d_\ell$ and free vars $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$
Let $F : \mathbb{R}^{d_1+\cdots+d_\ell} \to \mathbb{R}^d$ be a function. Then,

$$\varphi(\mathbf{x}) = F(\varphi_1, \ldots, \varphi_\ell)$$

is a higher-order MPNN of dim d and free vars $\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_\ell$

Higher-order MPNNs: Function Application

Function application: Syntax

Let $\varphi_1(\mathbf{x}_1), \ldots, \varphi_\ell(\mathbf{x}_1)$ be higher-order MPNNs of dim $d_1, \ldots, d_\ell$ and free vars $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$
Let $F : \mathbb{R}^{d_1+\cdots+d_\ell} \to \mathbb{R}^d$ be a function. Then,

$$\varphi(\mathbf{x}) = F(\varphi_1, \ldots, \varphi_\ell)$$

is a higher-order MPNN of dim d and free vars $\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_\ell$

Semantics

$$\xi_\varphi : (v_1, \ldots, v_p) \mapsto F\Big(\xi_{\varphi_1}(v_1, \ldots, v_p), \ldots, \xi_{\varphi_\ell}(v_1, \ldots, v_p)\Big)$$

Linear algebra
Activation functions
Anything you want…

Higher-order MPNNs: Aggregation

Aggregation: Syntax

Let $\varphi_1(\mathbf{x}_1, \mathbf{x}_2)$ and $\varphi_2(\mathbf{x}_1, \mathbf{x}_2)$ be higher-order MPNNs of dim d_1 and d_2 and free vars $\mathbf{x}_1, \mathbf{x}_2$. Let Θ be a function mapping bags of vectors in $\mathbb{R}^{d_1}$ to a vector in $\mathbb{R}^{d}$. Then,

$$\varphi(\mathbf{x}_1) = \mathsf{agg}^{\Theta}_{\mathbf{x}_2}[\varphi_1 \mid \varphi_2]$$

is a higher-order MPNN of dim d and free vars $\mathbf{x}_1$

Higher-order MPNNs: Aggregation

Aggregation: Syntax

Let $\varphi_1(\mathbf{x_1}, \mathbf{x_2})$ and $\varphi_2(\mathbf{x}_1, \mathbf{x}_2)$ be higher-order MPNNs of dim d_1 and d_2 and free vars $\mathbf{x}_1, \mathbf{x}_2$. Let Θ be a function mapping bags of vectors in $\mathbb{R}^{d_1}$ to a vector in $\mathbb{R}^d$. Then,

$$\varphi(\mathbf{x}_1) = \mathsf{agg}^{\Theta}_{\mathbf{x}_2}[\varphi_1 \mid \varphi_2]$$

is a higher-order MPNN of dim d and free vars $\mathbf{x}_1$

Semantics

$$\xi_\varphi : \mathbf{v} \mapsto \theta\Big(\Big\{\!\!\Big\{ \xi_{\varphi_1}(\mathbf{v}, \mathbf{w}) \mid \xi_{\varphi_2}(\mathbf{v}, \mathbf{w}) \neq \mathbf{0} \Big\}\!\!\Big\} \Big)$$

Higher-order MPNNs: Aggregation

Θ is e.g., summation and $\varphi_2(x, y) := E(x, y)$ and $\varphi_1(x, y) := \mathbf{1}[y = y]$
We can count degrees as follows:

$$\varphi(x) = \mathsf{agg}_y^{\mathsf{sum}}[\mathbf{1}[y = y] \mid E(x, y)]$$

Expressive Power of k-MPNNs

A higher-order MPNN is called a k-MPNN if it uses at most k variables.

k-MPNNs=class of k-MPNN

G. and J. Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)

Expressive Power of k-MPNNs

A higher-order MPNN is called a k-MPNN if it uses at most k variables.

k-MPNNs=class of k-MPNN

Theorem (G. And Reutter 2022)

$$\rho(k - \mathrm{MPNNs}) = \rho(k\text{-WL})$$

G. and J. Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)

Expressive Power of k-MPNNs

A higher-order MPNN is called a k-MPNN if it uses at most k variables.

k-MPNNs=class of k-MPNN

Theorem (G. And Reutter 2022)

$$\rho(k - \text{MPNNs}) = \rho(k\text{-WL})$$

Take away: Bounding architectures is easy!!

Just write your architecture as higher-order MPNNs

Count variables

G. and J. Reutter: *Expressiveness and approximation properties of graph neural networks* (2022)

We end with some examples …

MPNNs

We define $\varphi^{(0)}(x_1) := \mathbf{1}[x_1 = x_1]$
Then for $t > 0$, we get

$$\varphi^{(t)}(x_1) := \mathsf{Upd}^{(t)}\left(\varphi^{(t-1)}(x_1), \mathsf{agg}^{\Theta^{(t)}}_{x_2}\left[\varphi^{(t-1)}(x_2) \mid E(x_1, x_2)\right]\right)$$

For readout layer, we get

$$\varphi := \mathsf{agg}^{\Theta}_{x_1}\left[\varphi^{(L)}(x_1) \mid \mathbf{1}[x_1 = x_1]\right]\Big)$$

2 variables $\mapsto$ 1-WL

MPNNs: Gilmer et al.: *Neural message passing for quantum chemistry* (2017)

Graph Convolutional Networks

Use $D^{-1/2}(I + A)D^{-1/2}$ as propagation matrix

$\varphi(x_1) := F(\mathsf{agg}^{\mathsf{sum}}_{x_2}[\mathbf{1}[x_2 = x_2] \mid E(x_1, x)])$ with $F : \mathbb{R} \to \mathbb{R} : x \mapsto \dfrac{1}{\sqrt{1+x}}$

We can use $\psi(x_1, x_2) := \times\left(\times\left(\varphi(x_1), +\left(\mathbf{1}[x_1 = x_2], E(x_1, x_2)\right)\right), \varphi(x_2)\right)$ in the MPNN expressions from the previous slide.

2 variables $\mapsto$ 1-WL

GCN: Kipf and Welling: *Semi-supervised classification with graph convolutional networks* (2017)

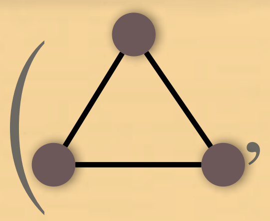
Simplified GNNs

- Uses path information $\mathbf{A}^p\mathbf{F}^{(0)}$ in a single layer.
- For $p = 3$ and for $\varphi^{(0)}(x_1)$ initial feature:
- $\psi(x_1) := \mathsf{agg}^{\mathsf{sum}}_{x_2}\left[\mathsf{agg}^{\mathsf{sum}}_{x_1}\left[\mathsf{agg}^{\mathsf{sum}}_{x_2}\left[\varphi^{(0)}(x_2) \mid E(x_1, x_2)\right] \mid E(x_2, x_1)\right] \mid E(x_1, x_2)\right]$

2 variables $\mapsto$ 1-WL

Wu et al. :*Simplifying Graph Convolutional Networks* (2019)

Subgraph count GNNs

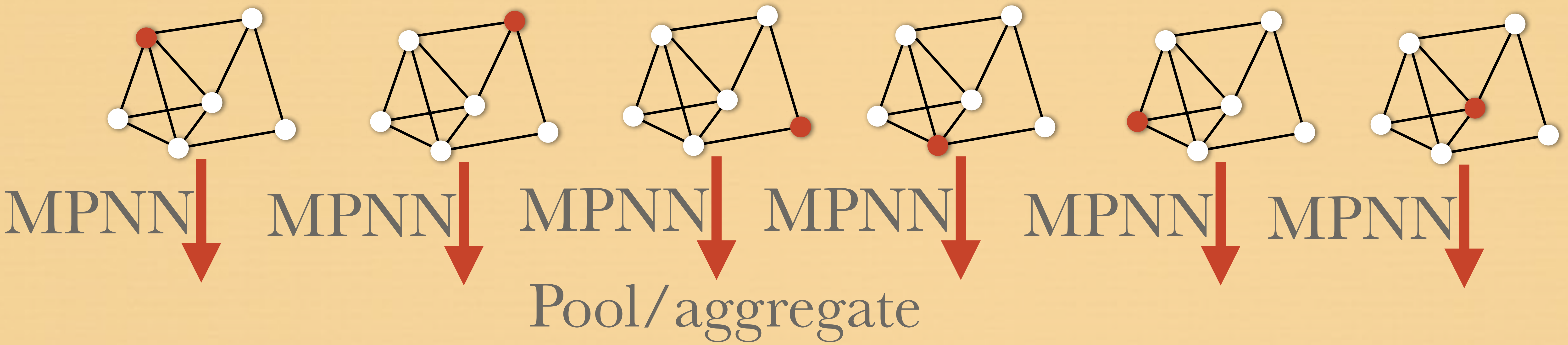
- Use count of subgraphs to augment MPNNs
  - homomorphism count $\mathsf{hom}(P^r, G^v)$ for rooted motif P,
  - subgraph iso count $\mathsf{sub}(P^r, G^v)$ for rooted motif P
- If motif has tree width k then $\mathsf{hom}(P^r, G^v)$ can be computed using k+1 variables.
- For example, $(G, v) \mapsto \mathsf{hom}\Big($  $, G^v\Big)$ can be expressed as

$$\varphi(x_1) := \sum_{x_2} \sum_{x_3} E(x_1, x_2)E(x_1, x_3)E(x_2, x_3)(\mathbf{1}[x_1 = x_1] - \mathbf{1}[x_1 = x_2])$$
$$(\mathbf{1}[x_1 = x_1] - \mathbf{1}[x_1 = x_3])(\mathbf{1}[x_1 = x_1] - \mathbf{1}[x_2 = x_3])$$

k+1 variables $\mapsto$ k-WL

Bouritsas et al.: *Improving graph neural network expressivity via subgraph isomorphism counting* (2020)
Barceló et al.: *Graph neural networks with local graph parameters.* (2021)

Subgraph GNNs: vertices

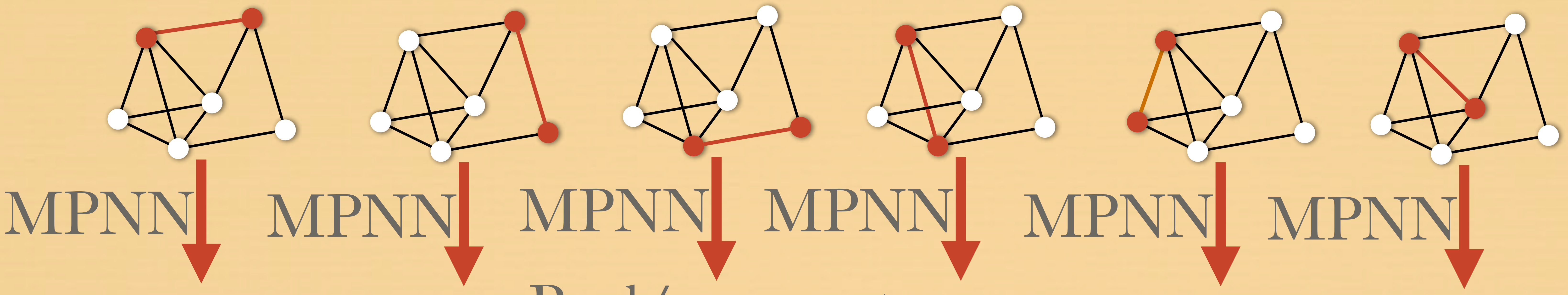


$$\varphi^{(0)}(x_1, x_2) := \mathbf{1}[x_1 = x_2]$$
$$\varphi^{(t)}(x_1, x_2) := \mathsf{Upd}^{(t)}\Big(\varphi^{(t-1)}(x_1, x_2), \mathsf{agg}^{\Theta}_{x_3}[\varphi^{(t-1)}(x_1, x_3) \mid E(x_2, x_3)]\Big)$$

3 variables $\mapsto$ 2-WL

Bevilacqua et al: *Equivariant subgraph aggregation network* (2022)
Cotta et al.: *Reconstruction for powerful graph representations* (2021)
Bevilacqua et al.: *Understanding and extending subgraph GNNs by rethinking their symmetries* (2022)
Huang et al.: *Boosting the cycle counting power of graph neural networks with I2-GNNs* (2022)
Papp et al.: *DropGNN: Random dropouts increase the expressiveness of graph neural networks.* (2021)
Qian et al.: *Ordered subgraph aggregation networks.* (2022)
You et al.: *Identity-aware graph neural networks.* (2021)
Zhang and P. Li. *Nested graph neural networks* (2021)
Zhao et al.: *From stars to subgraphs: Uplifting any GNN with local structure awareness* (2022)

Subgraph GNNs: edges



Pool/aggregate

$$\varphi^{(t)}(x_1, x_2, x_3) := \mathsf{Upd}^{(t)}\Big(\varphi^{(t-1)}(x_1, x_2, x_3), \mathsf{agg}^{\Theta}_{x_4}[\varphi^{(t-1)}(x_1, x_2, x_4) \mid E(x_3, x_4)]\Big)$$

4 variables $\mapsto$ 3-WL

Bevilacqua et al: *Equivariant subgraph aggregation network* (2022)
Cotta et al.: *Reconstruction for powerful graph representations* (2021)
Bevilacqua et al.: *Understanding and extending subgraph GNNs by rethinking their symmetries* (2022)
Huang et al.: *Boosting the cycle counting power of graph neural networks with I2-GNNs* (2022)
Papp et al.: *DropGNN: Random dropouts increase the expressiveness of graph neural networks.* (2021)
Qian et al.: *Ordered subgraph aggregation networks.* (2022)
You et al.: *Identity-aware graph neural networks.* (2021)
Zhang and P. Li. *Nested graph neural networks* (2021)
Zhao et al.: *From stars to subgraphs: Uplifting any GNN with local structure awareness* (2022)

Conclusion

- Takes a bit of practice but easy to get bounds
- Not guaranteed that these bounds are tight: depends on your programming skills in order to reduce number of variables.
- No lower bounds.

Please use it to get bounds!

And cite the ICLR paper :-)