Recurrent GNNs: The Power of Iteration

ESSAI 2026 course

Floris Geerts[1] Jonni Virtema[2]

[1] University of Antwerp, BE
[2] University of Glasgow, UK

Version of 6th July 2026

This course covers basics of graph learning, how theoretical limits are proven for (non-recurrent) GNNs, and what changes when recurrence is introduced.

Organisational information about the course

Material for the course:

- Lecture notes
- This slide set.
- Webpage: `http://www.virtema.fi/essai26.html`

About the lecturers

Floris Geerts (University of Antwerp)
Research Interests: Database Theory, Graph and Relational Learning, Quantum stuff.
`https://fgeerts.github.io/`

Jonni Virtema (University of Glasgow)
Research Interests: Logical Foundations of Neural Networks, Complexity Theory, Finite Model Theory, Temporal Logics for Hyperproperties.
`http://www.virtema.fi/`

Lecture 1: Basics of graph learning, GNNs, and notions of expressivity

Part I: Basics of Graph Learning

- What are we learning: Properties of graph structured data
- Constraints on learning: Invariants that any learned function must obey (e.g., permutation invariance).
- What is the model: Graph Neural Networks (GNNs) – built to satisfy these invariants.

Graphs

- We work with finite undirected graphs whose nodes carry labels.
- Let X be a set. An X-labeled graph is a triple

$$G = (N, E, g),$$

  where N is a finite set of nodes, $E \subseteq N \times N$ is the edge (symmetric, antireflexive) relation, and

$$g \colon N \to X$$

  assigns a label to each node.
- The map g is the node-labeling function. If $X = \mathbb{R}^d$, then g is also called a feature map.
- We write $\mathcal{G}[X]$ for the class of all finite X-labeled undirected graphs.

Graph data in the WILD


Event Graphs


Image credit: SalientNetworks
Computer Networks


Disease Pathways


Image credit: Wikipedia
Food Webs


Image credit: Pinterest
Particle Networks


Image credit: visitlondon.com
Underground Networks


Image credit: Medium
Social Networks

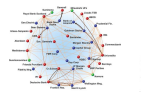


Image credit: Science
Economic Networks


Image credit: Lumen Learning
Communication Networks


Citation Networks


Image credit: Missoula Current News
Internet

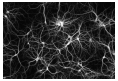


Image credit: The Conversation
Networks of Neurons


Image credit: Maximilian Nickel et al
Knowledge Graphs

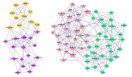


Image credit: eea.wustl.edu
Regulatory Networks


Image credit: math.hws.edu
Scene Graphs


Image credit: ResearchGate
Code Graphs

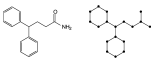


Image credit: MDPI
Molecules


Image credit: Wikipedia
3D Shapes

Basic graph concepts

Let $G = (N, E, g)$ be an X-labeled graph.

- For $n \in N$, the neighbourhood of n is

$$E(n) := \{m \in N \mid (n, m) \in E\}.$$

- The corresponding degree is

$$\deg(n) := |E(n)|.$$

- A pointed graph is a pair (G, n), where $n \in N$.

Graph learning tasks

- A node-level task predicts a property of each node:

$$n \mapsto \text{label of } n,$$

  e.g., classifying users in a social network.
- An edge-level task predicts a property of a pair of nodes:

$$(m, n) \mapsto \text{label of } (m, n),$$

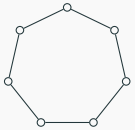
  such as whether an edge exists (*link prediction*) or its type (*edge classification*).
- A graph-level task predicts a property of the whole graph:

$$G \mapsto \text{label of } G,$$

  e.g., predicting a molecule's toxicity.
- In these lectures, we mainly focus on node classifiers mapping nodes to a finite set of labels.

Constraints on graph learning

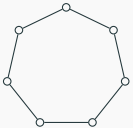
- In graph learning, we want to learn properties of graphs that should not depend on their representation.
- Intuitively: If two graphs look the same they should be treated the same!
- Example: Leader selection
  - In an odd linear path , which nodes could be selected as leaders?
  - In a cycle



which node set could be selected?

Constraints on graph learning

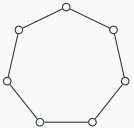
- In graph learning, we want to learn properties of graphs that should not depend on their representation.
- Intuitively: If two graphs look the same they should be treated the same!
- Example: Leader selection
  - In an odd linear path  , only the middle node can be distinguished from all others.
  - In a cycle

    which node set could be selected?

Constraints on graph learning

- In graph learning, we want to learn properties of graphs that should not depend on their representation.
- Intuitively: If two graphs look the same they should be treated the same!
- Example: Leader selection
  - In an odd linear path  ,
  - In a cycle



only two sets—empty set and all nodes—can be distinguished.

  - Why? Each node looks the identical from the node's perspective (automorphisms)!

Invariance: The formal notion capturing the constraints on graph learning

- Computations on graphs should not depend on arbitrary choices of representation.
  - In particular, node identifiers and the storage order of nodes or edges should not affect the result.
  - Graph isomorphisms formalize when two representations describe the same labeled graph.
- A graph-level function should assign the same value to isomorphic graphs.
  - This is called invariance.

Equivalence relations

- Graph learning deals with multiple notions of similarity captured by an equivalence relation.
- A binary relation $\sim \subseteq \mathcal{G}[X] \times \mathcal{G}[X]$ on labelled graphs is an equivalence relation, if it is
  - reflexive: $G \sim G$ for all labelled graphs $G \in \mathcal{G}[X]$,
  - symmetric: if $G \sim G'$ then $G' \sim G$, and
  - transitive: if $G_1 \sim G_2$ and $G_2 \sim G_3$ then $G_1 \sim G_3$.
- Examples of equivalence relations: equality, equicardinality, graph isomorphism.
- Equivalence relation for pointed graphs (or any base set) is defined analogously.

Graph isomorphisms

Let $G = (N, E, g)$ and $G' = (N', E', g')$ be X-labeled graphs.

- An isomorphism $f : G \to G'$ is a bijection $f : N \to N'$ such that, for all $m, n \in N$,

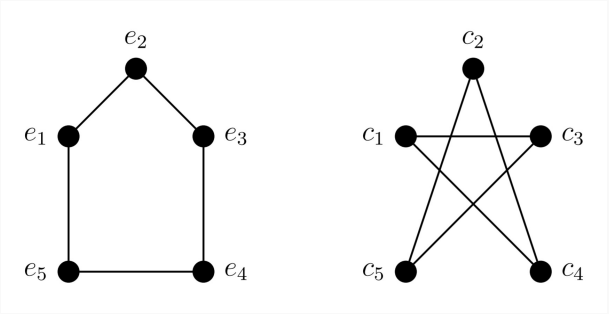
$$(m, n) \in E \iff (f(m), f(n)) \in E',$$

  and, for all $n \in N$,

$$g'(f(n)) = g(n).$$

- An isomorphism preserves both the edge structure and the node labels.
- We write $G \cong G'$ if there exists an isomorphism between G and G'.
- Analogously, $(G, n) \cong (G', n')$ for pointed graphs, for which $f(n) = n'$.
- Easy to check that $\cong$ is an equivalence relation on labelled graphs.

Graph isomorphism: Example



Graph invariants

Let Y be a set, $\sim_0$ be an equivalence relation on labelled graphs, and $\sim_1$ be an equivalence relation on labelled pointed graphs.

- A $\sim_0$-invariant is a function $\xi\colon \mathcal{G}[X] \to Y$ such that

  $$\xi(G) = \xi(G'),$$

  whenever $G \sim_0 G'$.
- A $\sim_1$-invariant assigns to every graph $G = (N, E, g)$ a function

  $$\xi_G\colon N \to Y$$

  that is invariant under $\sim_1$; that is

  $$\xi_G(n) = \xi_{G'}(n'),$$

  whenever $(G, n) \sim_1 (G', n')$.
- When $\sim_0$ and $\sim_1$ refer to graph isomorphism, we obtain the concepts of graph invariants and node invariants.

Graph invariants: examples

Let $G = (N, E, g)$ be an X-labeled graph.

- Examples of graph-level invariants are

$$G \mapsto |N|, \qquad G \mapsto |E|, \qquad G \mapsto \{\{g(n) \mid n \in N\}\}.$$

  where $\{\{\ \}\}$ denotes a multiset or bag.

- Examples of node-level invariants are

$$n \mapsto \deg(n) \qquad n \mapsto g(n).$$

- Non-examples are quantities depending on node identifiers or on the order in which nodes are stored.
- A cycle of length n has only two node-level invariants $\xi \colon N \to \mathbb{B}$: select all and select none.

The graph learning setting

- In graph learning, we are given training data consisting of graphs and target labels.
- For a graph-level task, the training data has the form

$$(G_1, y_1), \ldots, (G_r, y_r),$$

  where y_j is the target label for the graph G_j.
- For a node-level task, the training data has the form

$$((G_1, n_1), y_1), \ldots, ((G_r, n_r), y_r),$$

  where y_j is the target label for the graph G_j and node $n_j \in N_j$.
- The goal is to predict labels for new graphs or new nodes.

Why graph neural networks?

- Graph learning is not ordinary vector learning: the same graph can be represented using many different node orders.
- Learned functions should be invariant.
- Graph neural networks are designed so that these constraints are built into the architecture.
- They compute node representations by repeatedly combining

  the current representation of a node with representations of its neighbours.

- This gives a principled model for learning from node labels/features together with graph structure.

What is learned?

- As we will see graph neural networks (GNN) contain parameters, denoted by $\boldsymbol{\theta}$.
- These are typically weights and biases of neural networks used inside the GNN layers.
- The graph itself is not learned: the nodes, edges, and input features are the data.
- Training chooses $\boldsymbol{\theta}$ by minimizing a loss function on the training data.
- In expressivity theory, we ask a different question: which functions can be represented by some choice of parameters?

The basic idea of message passing

- Each node starts with an initial feature vector.
- If initial labels are not in $\mathbb{R}^d$, then an initial encoding is needed. For example, a one-hot encoding of discrete labels.
- In each layer, every node collects information from its neighbours.
- This neighbour information is combined with the node's current feature.
- After ℓ layers, the feature of a node depends on information up to distance ℓ.
- Thus, GNNs compute learned node representations, and possibly graph representations.

Graph Neural Networks: A single layer

- A GNN is built from layers.
- A layer $L_{\boldsymbol{\theta}}$ maps an $\mathbb{R}^p$-labeled graph

$$G = (N, E, g)$$

to an $\mathbb{R}^q$-labeled graph

$$L_{\boldsymbol{\theta}}(G) = (N, E, g').$$

- The nodes and edges stay fixed; only the node features are updated.
- For $n \in N$, the updated feature is

$$g'(n) \coloneqq \mathrm{Comb}_{\boldsymbol{\theta}}\Big(g(n), \mathrm{Agg}_{\boldsymbol{\theta}}\big(\{\{g(m) \mid m \in E(n)\}\}\big)\Big).$$

Here,

$$\mathrm{Agg}_{\boldsymbol{\theta}} \colon \mathcal{M}(\mathbb{R}^p) \to \mathbb{R}^h, \qquad \mathrm{Comb}_{\boldsymbol{\theta}} \colon \mathbb{R}^p \times \mathbb{R}^h \to \mathbb{R}^q.$$

$\mathcal{M}(\mathbb{R}^p)$ is the set of finite multisets of elements of the set $\mathbb{R}^p$.

Aggregate–Combine GNNs (AC-GNNs)

- The name AC refers to the two steps in one message-passing layer:

  Aggregate neighbour features $+$ Combine with own feature.

- For $n \in N$:

$$g'(n) := \underbrace{\mathrm{Comb}_{\boldsymbol{\theta}}\Big(g(n), \underbrace{\mathrm{Agg}_{\boldsymbol{\theta}}\big(\{\{g(m) \mid m \in E(n)\}\}\big)}_{\text{Aggregate}}\Big)}_{\text{Combine}}.$$

- Thus an AC layer first summarizes the neighbourhood of n, then updates the feature of n using this summary.

Graph Neural Networks: stacking layers

- A GNN consists of layers
$$L^{(1)}_{\boldsymbol{\theta}_1}, L^{(2)}_{\boldsymbol{\theta}_2}, \ldots, L^{(\ell)}_{\boldsymbol{\theta}_\ell}.$$
- Given an input graph
$$G_0 = (N, E, g_0), \qquad g_0 \colon N \to \mathbb{R}^{p_0},$$
the layers recursively define
$$G_i = (N, E, g_i) := L^{(i)}_{\boldsymbol{\theta}_i}(G_{i-1})$$
for $i = 1, \ldots, \ell$.
- The output dimension of $L^{(i)}_{\boldsymbol{\theta}_i}$ must match the input dimension of $L^{(i+1)}_{\boldsymbol{\theta}_{i+1}}$.
- The final node representation is
$$n \mapsto g_\ell(n).$$

Graph Neural Networks: readout

- For node-level tasks, the output is computed from the final node features:

$$n \mapsto g_\ell(n).$$

- For graph-level tasks, one adds a readout function

$$\mathrm{ReadOut}_{\boldsymbol{\theta}} \colon \mathcal{M}(\mathbb{R}^{p_\ell}) \to \mathbb{R}^q.$$

- The graph-level output is

$$\mathrm{ReadOut}_{\boldsymbol{\theta}}\big(\{\{g_\ell(n) \mid n \in N\}\}\big).$$

- For classification, the real-valued output is mapped to a finite label set $Y = \{1, \ldots, q\}$, for example by

$$\mathrm{Classify}(z) = \arg\max_{i \in Y} z_i, \qquad z \in \mathbb{R}^q.$$

Training a GNN

- A GNN with parameters $\boldsymbol{\theta}$ defines a prediction function $F_{\boldsymbol{\theta}}$.
- Given graph-level training data

$$(G_1, y_1), \ldots, (G_r, y_r),$$

the parameters are chosen by minimizing an empirical loss

$$\min_{\boldsymbol{\theta}} \frac{1}{r} \sum_{j=1}^{r} \ell\big(F_{\boldsymbol{\theta}}(G_j), y_j\big).$$

- The learned function should generalize to graphs not seen during training.
- In the rest of the lecture, we abstract away from training and focus on expressive power.

GNN layers are invariant

Let $G = (N, E, g)$ and $G' = (N', E', g')$ be isomorphic via $f : G \to G'$.

- Since f preserves edges,
$$f(E(n)) = E'(f(n)).$$
- Since f preserves labels,
$$g'(f(m)) = g(m).$$
- Therefore the multisets of neighbour features agree:
$$\{\{g(m) \mid m \in E(n)\}\} = \{\{g'(m') \mid m' \in E'(f(n))\}\}.$$
- Hence every message-passing layer is node-invariant.

Simple AC-GNN layers

A common special case:

- Uses sum aggregation for Agg;
- Use feedforward neural networks for Comb.
- At layer i, the update is

$$g_i(n) = f_i\left(g_{i-1}(n) \mid \sum_{m \in E(n)} g_{i-1}(m)\right).$$

- Here $x \mid y$ denotes vector concatenation and f_i is a feedforward neural network (FNN).
- This type of GNNs will be used later in these lectures.

Feedforward neural networks

- An FNN is a function $f : \mathrm{R}^k \to \mathrm{R}^l$ built by composing layers of the form

$$z \mapsto \sigma(Wz + b),$$

  where W is a weight matrix, b a bias vector, and σ a (non-linear) activation function applied component-wise.
- A network with L layers computes

$$f(x) = \sigma_L\big(W_L\, \sigma_{L-1}(\cdots \sigma_1(W_1 x + b_1)\cdots) + b_L\big).$$

- Typical choices for σ: ReLU, sigmoid, tanh; the output layer often has no activation (or a task-specific one).

Boolean graph properties

Let $\mathbb{B} := \{0, 1\}$.

- A Boolean graph property is a graph invariant

$$P \colon \mathcal{G}[X] \to \mathbb{B}.$$

- Examples:

$$\begin{aligned} P_{\mathrm{red}}(G) &= 1 && \text{iff } G \text{ contains a red node,} \\ P_{\mathrm{red}\to\mathrm{blue}}(G) &= 1 && \text{iff some red node has a blue neighbour.} \\ P_{\mathrm{conn}}(G) &= 1 && \text{iff } G \text{ is connected,} \\ P_{\Delta}(G) &= 1 && \text{iff } G \text{ contains a triangle.} \end{aligned}$$

Node classifiers

Let $\mathbb{B} := \{0, 1\}$.

- A node classifier is a label transformer

$$C \colon \mathcal{G}[X] \to \mathcal{G}[\mathbb{B}].$$

- Thus, for every input graph $G = (N, E, g)$,

$$C(G) = (N, E, c_G), \qquad c_G \colon N \to \mathbb{B}.$$

- Equivalently, C selects a set of nodes

$$C_G := \{n \in N \mid c_G(n) = 1\}.$$

Node classifiers: examples

$$
\begin{aligned}
c_{\mathrm{red}}(G, n) = 1 &\iff g(n) = \mathrm{red}, \\
c_{\text{blue-nbr}}(G, n) = 1 &\iff \exists m \in E(n) : g(m) = \mathrm{blue}, \\
c_{\text{reach-blue}}(G, n) = 1 &\iff n \text{ can reach some blue node}, \\
c_{\mathrm{cycle}}(G, n) = 1 &\iff n \text{ lies on a cycle}.
\end{aligned}
$$

Why Boolean properties?

- Boolean properties are a convenient way to study expressive power.
- Instead of asking whether a model computes a complicated output, we ask whether it can answer a yes/no question.
- This already captures many important tasks:

Does the graph contain a red node? Is this node on a cycle?

- Boolean graph properties correspond to graph-level tasks.
- Boolean node classifiers correspond to node-level tasks.
- Later, they will also match naturally with logical formulas.

From learning to expressivity

- We have seen that GNNs define functions that respect graph symmetries.
- Expressivity asks which invariant graph properties and node classifiers can be represented by such architectures.
- Boolean properties give a clean test language:

can the model answer this yes/no question?

- Local properties, such as seeing a blue neighbour, are natural for message passing.
- Global properties, such as reachability and cycle membership, reveal its limitations.
- Part 2 makes this precise using colour refinement.

Part 2: Limits of Graph Neural Networks

- How capabilities of GNNs can be measured?
  - Distinguising power.
  - Uniform and non-uniform expressivity.
- To what can we compare GNNs?
  - Graph algorithms (colour refinement).
  - Later: Logics

Node-feature maps

Fix input labels X and output labels Y. Let $\mathcal{G}[X]$ denote the class of graphs whose nodes are labelled by X.

- A node-feature map is a graph transformation

$$M \colon \mathcal{G}[X] \to \mathcal{G}[Y]$$

  that changes only the node labels, not the underlying graph.
- Thus, for every input graph $G = (N, E, g)$,

$$M(G) = (N, E, h_G), \qquad h_G \colon N \to Y.$$

- In words: M assigns a new output label $h_G(n)$ to each node $n \in N$.
- A class $\mathcal{M}$ is a set of such maps.
- To study the expressivity of a GNN architecture, we study the class of node-feature maps it can produce.

What does it mean to be powerful?

- There are several ways to ask how powerful a class $\mathcal{M}$ of node-feature maps is.
- The weakest question is about distinguishing power:

  can some map in $\mathcal{M}$ tell these two nodes apart?

- The strongest question is about uniform expressivity:

  can one fixed map in $\mathcal{M}$ compute this classifier on all graphs?

- Between them sits non-uniform expressivity: the map may depend on the size of the input, or even on the particular comparison.
- The phrase expressive power is used for all of these in the literature, so we must say which question we are asking.

How to measure the power of classes of node feature maps?

- Many ways to classify the computing power of a class $\mathcal{M}$ of feature maps.
- Coarsest: distinguishability
- Finest: uniform expressiveness
- Middle ground: non-uniform expressiveness

How to measure the power of classes of node feature maps?

- Many ways to classify the computing power of a class $\mathcal{M}$ of feature maps.
- Coarsest: distinguishability
  - Given two pointed graphs (G, n) and (H, m), we ask whether there exists a node-feature map $M \in \mathcal{M}$ that separates the pointed graphs, i.e., such that

$$M(G)(n) \neq M(H)(m).$$

  - The map may depend on the size (or any other information) of the two pointed graphs.
  - Distinguishability does not answer to the question which classifiers can be expressed by a feature map.
  - It can be used to show which classifiers cannot be expressed!
- Finest: uniform expressiveness
- Middle ground: non-uniform expressiveness

How to measure the power of classes of node feature maps?

- Many ways to classify the computing power of a class $\mathcal{M}$ of feature maps.
- Coarsest: distinguishability
- Finest: uniform expressiveness
  - Which classifiers can be computed by using a single feature map from $\mathcal{M}$.
  - The same parameters, architecture, and number of layers must work for all input graphs.
  - Uniform expressivity answers to the question what classifiers belong to the class $\mathcal{M}$.
- Middle ground: non-uniform expressiveness

How to measure the power of classes of node feature maps?

- Many ways to classify the computing power of a class $\mathcal{M}$ of feature maps.
- Coarsest: distinguishability
- Finest: uniform expressiveness
- Middle ground: non-uniform expressiveness
  - Different maps can be used for differet inputs; map selection may depend on graph size.
  - For example, one may allow a family of feature maps

    $$(M_s)_{s\in\mathbb{N}},$$

    where M_s only has to work on graphs of size s.
  - Then a node classifier C is non-uniformly expressible if

    $$M_{|N|}(G)(n) = C(G)(n)$$

    for every graph $G = (N, E, g)$ and every node $n \in N$.
  - This is weaker than uniform expressiveness, since the model may change with the graph size. In general $s \mapsto M_s$, need not be even computable!

How to measure the power of classes of node feature maps?

- Many ways to classify the computing power of a class $\mathcal{M}$ of feature maps.
- Coarsest: distinguishability
- Finest: uniform expressiveness
- Middle ground: non-uniform expressiveness
- In the literature, expressive power may refer to any of the above notions.

Distinguishing power: what can never be separated?

- A natural first question is:

  are there theoretical limits to what GNNs can distinguish?

- The first limit comes from graph isomorphism.
- If two pointed graphs are isomorphic,

$$(G, n) \cong (H, m),$$

  then any isomorphism-invariant node-feature map must treat n and m the same.
- In particular, if $f : G \to G$ is an automorphism, then n and $f(n)$ cannot be separated.
- So the first wall is symmetry:

  GNNs cannot break graph symmetries.

- The more interesting wall is colour refinement. That is where we go next.

Colour refinement: what does a node see?

- Imagine that every node repeatedly asks:

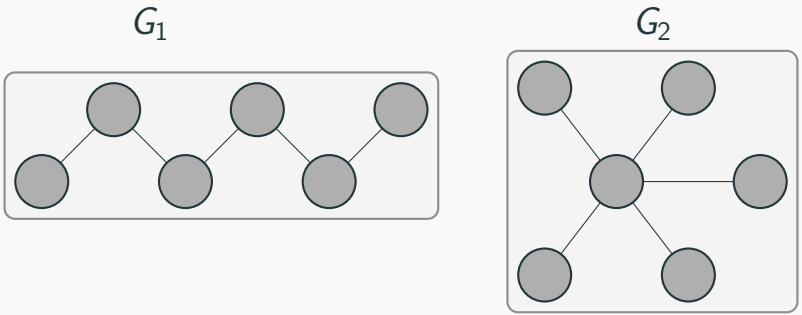
  what colour am I, and what colours do my neighbours have?

- Initially, a node only knows its own input label:

$$\chi_0(n) := g(n).$$

- After one round, it knows the multiset of labels around it.
- After two rounds, it knows how those neighbours themselves look.
- After t rounds, the colour $\chi_t(n)$ summarizes the depth-t view of n as seen by this message-passing process.

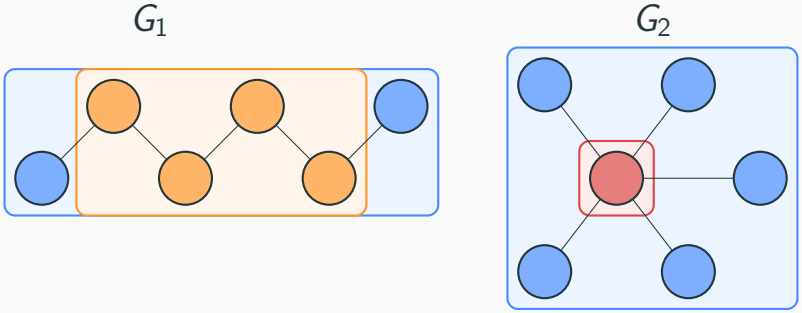
Colour Refinement (1-WL)



Round 0: one colour class

$$\chi_0(n) = \text{grey}$$

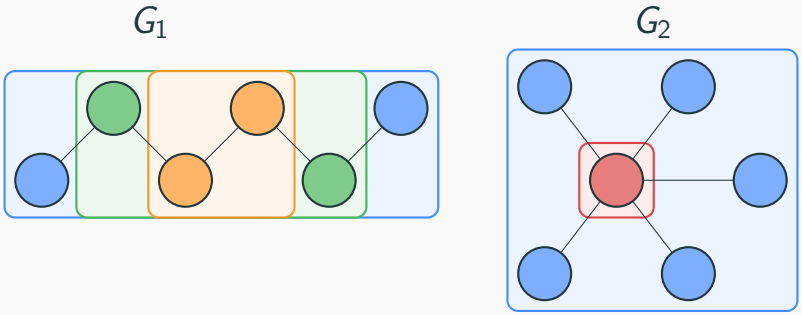
Colour Refinement (1-WL)



Round 1: split by degree

$$\chi_{t+1}(n) = \left(\chi_t(n), \{\!\{\chi_t(m) : m \in E(n)\}\!\}\right)$$

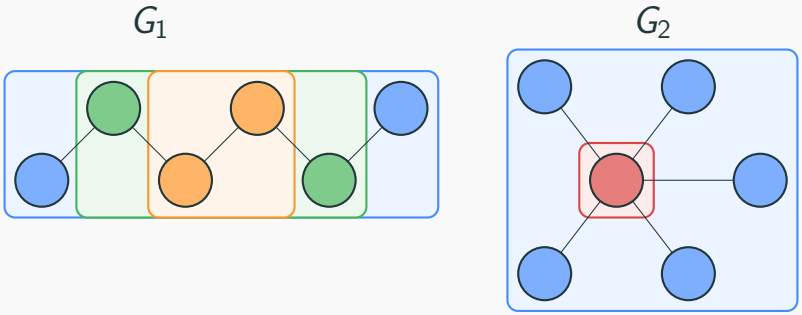
Colour Refinement (1-WL)



Round 2: colours propagate

$$\chi_{t+1}(n) = \big(\chi_t(n), \{\!\{\chi_t(m) : m \in E(n)\}\!\}\big)$$

Colour Refinement (1-WL)



Stable colouring

$$\chi_{t+1}(n) = \left(\chi_t(n), \{\!\{\chi_t(m) : m \in E(n)\}\!\}\right)$$

Colour refinement in one line

- The update rule is

$$\chi_{t+1}(n) \coloneqq \mathsf{hash}\Big(\chi_t(n), \{\{\chi_t(m) \mid m \in E(n)\}\}\Big).$$

- The hash is injective: two nodes get the same new colour exactly when they had
  - the same old colour, and
  - the same multiset of neighbour colours.
- So colour refinement never guesses. It separates nodes only when their local evidence is different.
- When no further split is possible, the colouring is stable.

Stable colouring

Let $\chi_t \colon N \to C_t$ be the colouring after round t.

- Each colouring χ_t induces a partition of N:

$$\Pi_t := \{\chi_t^{-1}(c) \mid c \in \chi_t(N)\}.$$

- Two nodes are in the same block of Π_t exactly when they have the same colour:

$$n \equiv_{\Pi_t} m \quad \iff \quad \chi_t(n) = \chi_t(m).$$

- Colour refinement stops once this partition no longer changes.
- Equivalently, it stops at the first T such that

$$\chi_T(n) = \chi_T(m) \quad \iff \quad \chi_{T+1}(n) = \chi_{T+1}(m)$$

  for all $n, m \in N$.
- The resulting stable colouring is denoted by χ_∞.

Colour histograms

Let $\chi\colon N \to C$ be a colouring of the nodes of a graph $G = (N, E, g)$.

- The colour histogram of χ records how often each colour occurs.
- Formally, it is the function

$$\mathrm{hist}_\chi\colon C \to \mathbb{N}$$

  defined by

$$\mathrm{hist}_\chi(c) := |\{n \in N \mid \chi(n) = c\}|.$$

- Equivalently, the colour histogram is the multiset of node colours

$$\{\{\chi(n) \mid n \in N\}\}.$$

- Thus two graphs have the same colour histogram at round t if they contain the same colours with the same multiplicities.

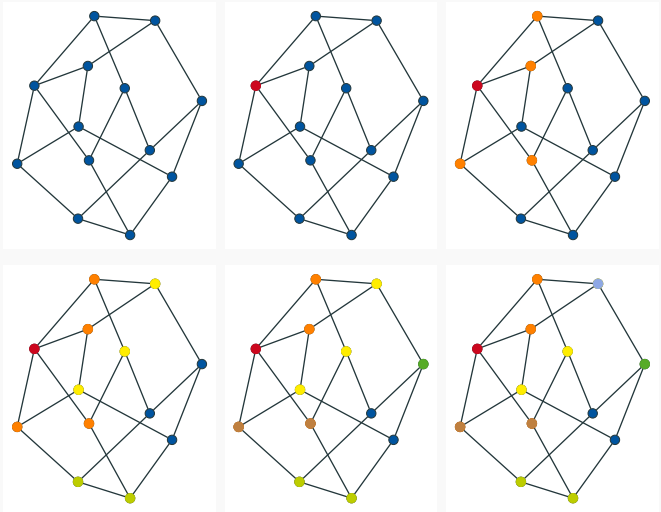
Graph isomorphism and colour refinement

- Colour refinement is invariant under graph isomorphism.
- If $G \cong G'$ via f, then for every round t,

$$\chi_t^G(n) = \chi_t^{G'}(f(n)).$$

- Hence, if colour refinement produces different colour histograms, then the graphs are not isomorphic.
- If the histograms agree, the test is inconclusive.
- Thus colour refinement is a useful but incomplete test for graph isomorphism.

Colour refinement: another example



GNNs and colour refinement

- Colour refinement and message-passing GNNs use the same basic pattern:

  old node information $+$ multiset of neighbour information.

- Colour refinement updates colours by

$$\chi_{t+1}(n) \coloneqq \mathsf{hash}\Big(\chi_t(n), \{\{\chi_t(m) \mid m \in E(n)\}\}\Big).$$

- A GNN layer updates features by

$$g_{t+1}(n) \coloneqq \mathsf{Comb}_t\Big(g_t(n), \mathsf{Agg}_t\{\{g_t(m) \mid m \in E(n)\}\}\Big).$$

- Thus colour refinement is a discrete analogue of message passing.

The punchline

A message-passing GNN cannot see more than colour refinement sees.

- This is not a statement about training.
- It is not a statement about the number of parameters.
- It is a structural statement about the form of message passing:

  a node repeatedly receives only a multiset summary of its neighbours.

Proposition 1 (Colour refinement and GNNs [Mor+19])

Let (G, n) and (H, m) be two pointed X-labeled graphs. Let χ_t denote the colouring after t rounds of colour refinement.

1. *For every t-layer message-passing GNN with node features g_t, we have*

$$\chi_t^G(n) = \chi_t^H(m) \quad \Longrightarrow \quad g_t^G(n) = g_t^H(m).$$

2. *Conversely, if colour refinement distinguishes the two pointed graphs after t rounds, that is,*

$$\chi_t^G(n) \neq \chi_t^H(m),$$

then there exists a t-layer message-passing GNN such that

$$g_t^G(n) \neq g_t^H(m).$$

Easy direction: GNNs are bounded by CR

- Let χ_t be the colour refinement colouring after t rounds.
- Let g_t be the node features computed by a t-layer GNN.
- The easy direction says:

$$\chi_t(n) = \chi_t(m) \quad \Longrightarrow \quad g_t(n) = g_t(m).$$

- Equivalently:

$$g_t(n) \neq g_t(m) \quad \Longrightarrow \quad \chi_t(n) \neq \chi_t(m).$$

- Hence a GNN cannot distinguish two nodes that colour refinement has not distinguished.

Proof of the easy direction

We prove by induction on t that

$$\chi_t(n) = \chi_t(m) \quad \Longrightarrow \quad g_t(n) = g_t(m).$$

- Base case $t = 0$: if $\chi_0(n) = \chi_0(m)$, then n and m have the same initial label, so

$$g_0(n) = g_0(m).$$

- Induction step: assume the claim holds for t.
- Suppose

$$\chi_{t+1}(n) = \chi_{t+1}(m).$$

- By definition of colour refinement,

$$\chi_t(n) = \chi_t(m)$$

and

$$\{\{\chi_t(u) \mid u \in E(n)\}\} = \{\{\chi_t(v) \mid v \in E(m)\}\}.$$

Proof of the easy direction continued

By the induction hypothesis, equal colours at round t imply equal GNN features.

- Therefore the neighbour-feature multisets agree:

$$\{\{g_t(u) \mid u \in E(n)\}\} = \{\{g_t(v) \mid v \in E(m)\}\}.$$

- Also,

$$g_t(n) = g_t(m).$$

- Since the same aggregation and combination functions are applied at every node,

$$g_{t+1}(n) = g_{t+1}(m).$$

- This proves the induction step.
- Hence every message-passing GNN is bounded by colour refinement.

Consequence

- If colour refinement gives two nodes the same stable colour, then no message-passing GNN can distinguish them:

$$\chi_\infty(n) = \chi_\infty(m) \quad \Longrightarrow \quad g_\ell(n) = g_\ell(m)$$

  for every number of layers ℓ.
- Therefore, message-passing GNNs are at most as expressive as colour refinement.
- In particular, every node classifier definable by such a GNN is constant on colour-refinement classes.

The converse direction

- The easy direction was:

$$\text{GNN distinguishes} \implies \text{CR distinguishes.}$$

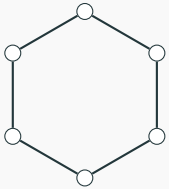
- The converse asks:

$$\text{CR distinguishes} \implies \text{some GNN distinguishes.}$$

- We show this later using logic!

Colour refinement cannot distinguish $2K_3$ from C_6

$G = 2K_3$



$H = C_6$

- Colour refinement does not distinguish $2K_3$ and C_6, even though they are not isomorphic.
- Neither can any GNN! Irregardless of number of layers or number of parameters or complexity of combine and aggregate functions..

Why colour refinement cannot see triangles

- Initially, all nodes have the same colour.
- In both graphs, every node has exactly two neighbours.
- Moreover, as long as all nodes have the same colour, every node sees the same multiset of neighbour colours:

$$\{\{\chi_t(m) \mid m \in E(n)\}\} = \{\{c_t, c_t\}\}.$$

- Hence all nodes receive the same colour at the next round:

$$\chi_{t+1}(n) = c_{t+1} \qquad \text{for every node } n.$$

- By induction, colour refinement never distinguishes nodes in $K_3 \sqcup K_3$ from nodes in C_6.

Consequence for GNNs

Define the node classifier

$$C_\Delta(G)(n) = 1 \quad \iff \quad n \text{ lies in a triangle.}$$

- In $K_3 \sqcup K_3$, every node satisfies $C_\Delta e(G)(n) = 1$.
- In C_6, every node satisfies $C_\Delta(H)(n) = 0$.
- But colour refinement assigns the same stable colour to all nodes in both graphs.
- Since message-passing GNNs are bounded by colour refinement, no standard message-passing GNN can express C_Δ.
- The problem is structural:

  message passing cannot tell whether two neighbours are connected to each other.

A different blind spot: information too far away

- The triangle example was not about distance. The triangle is right next to the node.
- Now consider a different limitation.
- After ℓ layers, a node can only receive information from nodes within distance ℓ.
- Therefore any fixed-depth GNN is local:

$$g_\ell(n) \text{ depends only on the labelled radius-}\ell \text{ neighbourhood of } n.$$

- So fixed-depth GNNs cannot express properties where the relevant witness may be arbitrarily far away.

Reachability is not fixed-depth

Fix a label p and define

$$C_{\mathrm{Reach}}(G)(n) = 1$$

iff there is a directed path from n to some node labelled p.

- For any fixed number of layers ℓ, choose two directed paths whose first ℓ steps from the start node look identical.
- In the first graph, a p-node occurs just beyond this visible horizon, at distance $\ell + 1$.
- In the second graph, no p-node is reachable.
- The start nodes have the same radius-ℓ view, so every ℓ-layer GNN gives them the same output.
- Hence no fixed-depth GNN can express general reachability.

Reachability needs iteration

Reachability is naturally computed by an iterative process.

- Start with the target nodes:

$$R_0 := \{n \mid p \in g(n)\}.$$

- Then repeatedly add predecessors:

$$R_{t+1} := R_t \cup \{n \mid E(n) \cap R_t \neq \emptyset\}.$$

- The process may require as many rounds as the length of the longest relevant path.
- Thus reachability is not a fixed-depth property.

From distinguishing power to expressive power (a)

- Distinguishing two fixed graphs can be easier than expressing one classifier uniformly on all graphs.
- For example, if G and G' are paths of different lengths, then some GNN can separate them.
- But as the paths become longer, the separating GNN may need more layers.
- This matters because uniform expressivity requires one fixed depth.
- Therefore: separating every finite example individually is not the same as computing one global classifier.

From distinguishing power to expressive power (b)

- Easy exercise: If G and G' are paths of different lengths n and m, there exists a GNN that separates G and G'.
  - Caveat: When n and m grow bigger, the separating GNN requires more layers.
- Separating power of l-layer GNNs is limited to that of l-round colour refinement.
  - Consequence: Any graph property that can be expressed with a GNN is invariant under l-round colour refinement, for large enough l!

From distinguishing power to expressive power (c)

- Fix l, no l-layer GNN can separate G, a path of length $3l + 2$, from G', a disjoint union of a path of $2l + 2$ and a cycle of length l.
  - Proof: The l-neighbourhoods of the l middle points of G are indistinguishable from the cycle points of G'. This can be formalised using l-round CR.
- Consequence: No fixed layer GNN can decide whether a graph has a cycle.

Two kinds of limitations

- Some properties fail because fixed-depth GNNs are local.
- Reachability is an example: the relevant information may be arbitrarily far away.
- Other properties fail because they are not visible to colour refinement, as we have seen before.

Reachability vs. cycle membership

- Reachability illustrates a locality limitation: fixed-depth GNNs cannot follow paths of unbounded length.
- Cycle membership illustrates a colour-refinement limitation.
- CR sees the unfolding of the graph around a node.
- It does not know whether a path eventually returns to the same node.
- Hence even very small cycles can be invisible to CR and to CR-bounded GNNs.

Takeaways

- Graphs provide a natural data model for relational data.
- GNNs learn node representations by message passing.
- Message-passing GNNs are invariant and define valid node classifiers.
- Their expressive power is closely tied to colour refinement.
  - They have the same distinguishing power!
  - Properties not invariant under colour refinement, such as Triangle, cannot be expressed by standard message-passing GNNs.
- Fixed-depth GNNs are local and cannot express general reachability.