

myExperience Surveys

The UNSW My Experience survey still open for 22T2, participation is highly encouraged.

“Please participate in the myExperience Survey and take the opportunity to share your constructive thoughts on your 2022 learning experience. Your contributions help your teachers and shape the future of education at UNSW.”

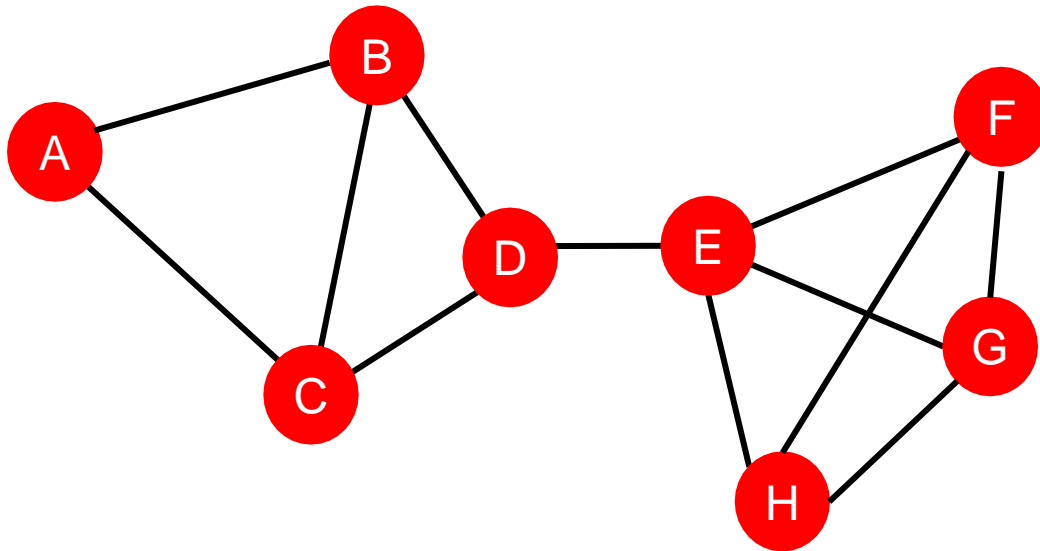
More information:

<https://www.student.unsw.edu.au/myexperience>

Graph-Level Features and Graph Kernels

Graph-level Features

- **Goal:** We want features that characterize the structure of an entire graph.
- **For example:**



Kernel Methods

- **Kernel methods** are widely-used for traditional ML for graph-level prediction.
- **Idea: Design kernels instead of feature vectors.**
- **A quick introduction to Kernels:**
 - Kernel $K(G, G') \in \mathbb{R}$ measures similarity b/w data
 - Kernel matrix $\mathbf{K} = \left(K(G, G') \right)_{G, G'}$ must always be positive semidefinite (i.e., has positive eigenvals)
 - There exists a feature representation $\phi(\cdot)$ such that $K(G, G') = \phi(G)^T \phi(G')$
 - Once the kernel is defined, off-the-shelf ML model, such as **kernel SVM**, can be used to make predictions.

Overview

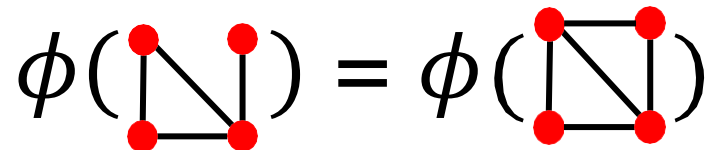
- **Graph Kernels:** Measure similarity between two graphs:
 - Graphlet Kernel [1]
 - Weisfeiler-Lehman Kernel [2]
 - Other kernels are also proposed in the literature (beyond the scope of this lecture)
 - Random-walk kernel
 - Shortest-path graph kernel
 - And many more...

1 Shervashidze, Nino, et al. "Efficient graphlet kernels for large graph comparison." Artificial Intelligence and Statistics. 2009.

2 Shervashidze, Nino, et al. "Weisfeiler-lehman graph kernels." Journal of Machine Learning Research 12.9 (2011).

Key Idea

- **Goal:** Design graph feature vector $\phi(G)$
- **Key idea:** Bag-of-Words (BoW) for a graph
 - **Recall:** BoW simply uses the word counts as features for documents (no ordering considered).
 - Naïve extension to a graph: **Regard nodes as words.**
 - Since both graphs have **4 red nodes**, we get the same feature vector for two different graphs...

$$\phi(\text{graph 1}) = \phi(\text{graph 2})$$


Key Idea

What if we use Bag of node degrees?

Deg1: ● Deg2: ● Deg3: ●

$$\phi(\text{triangle}) = \text{count}(\text{triangle with colored nodes}) = [1, 2, 1]$$

$$\phi(\text{square}) = \text{count}(\text{square with colored nodes}) = [0, 2, 2]$$



Obtains different features for different graphs!

- Both Graphlet Kernel and Weisfeiler-Lehman (WL) Kernel use **Bag-of-*** representation of graph, where ***** is more sophisticated than node degrees!

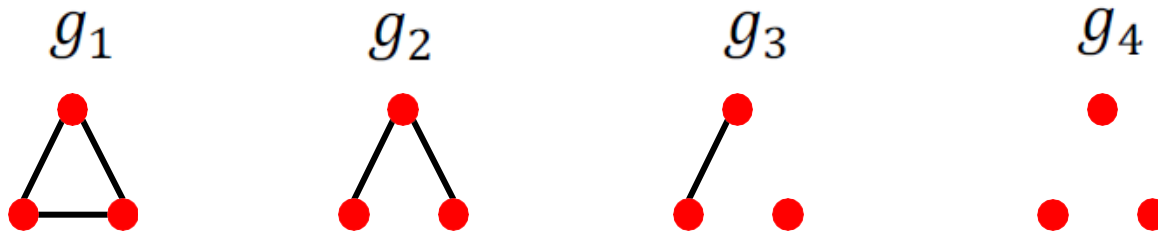
Graphlet Features

- **Key idea:** Count the number of different graphlets in a graph.
- **Note:** Definition of graphlets here is slightly different from node-level features.
- The two differences are:
 - Nodes in graphlets here do **not need to be connected** (allows for isolated nodes)
 - The graphlets here are not rooted.
 - Examples in the next slide illustrate this.

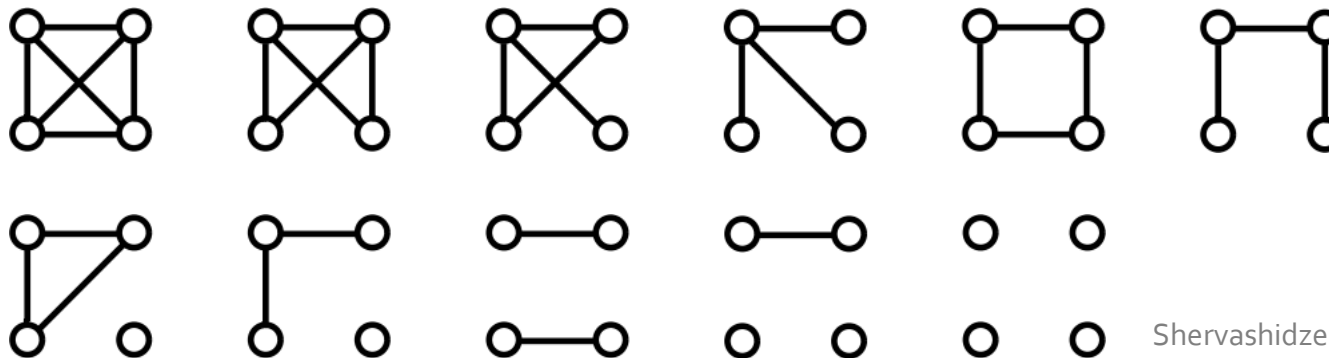
Graphlet Features

Let $\mathcal{G}_k = (g_1, g_2, \dots, g_{n_k})$ be a list of graphlets of size k .

- For $k = 3$, there are 4 graphlets.



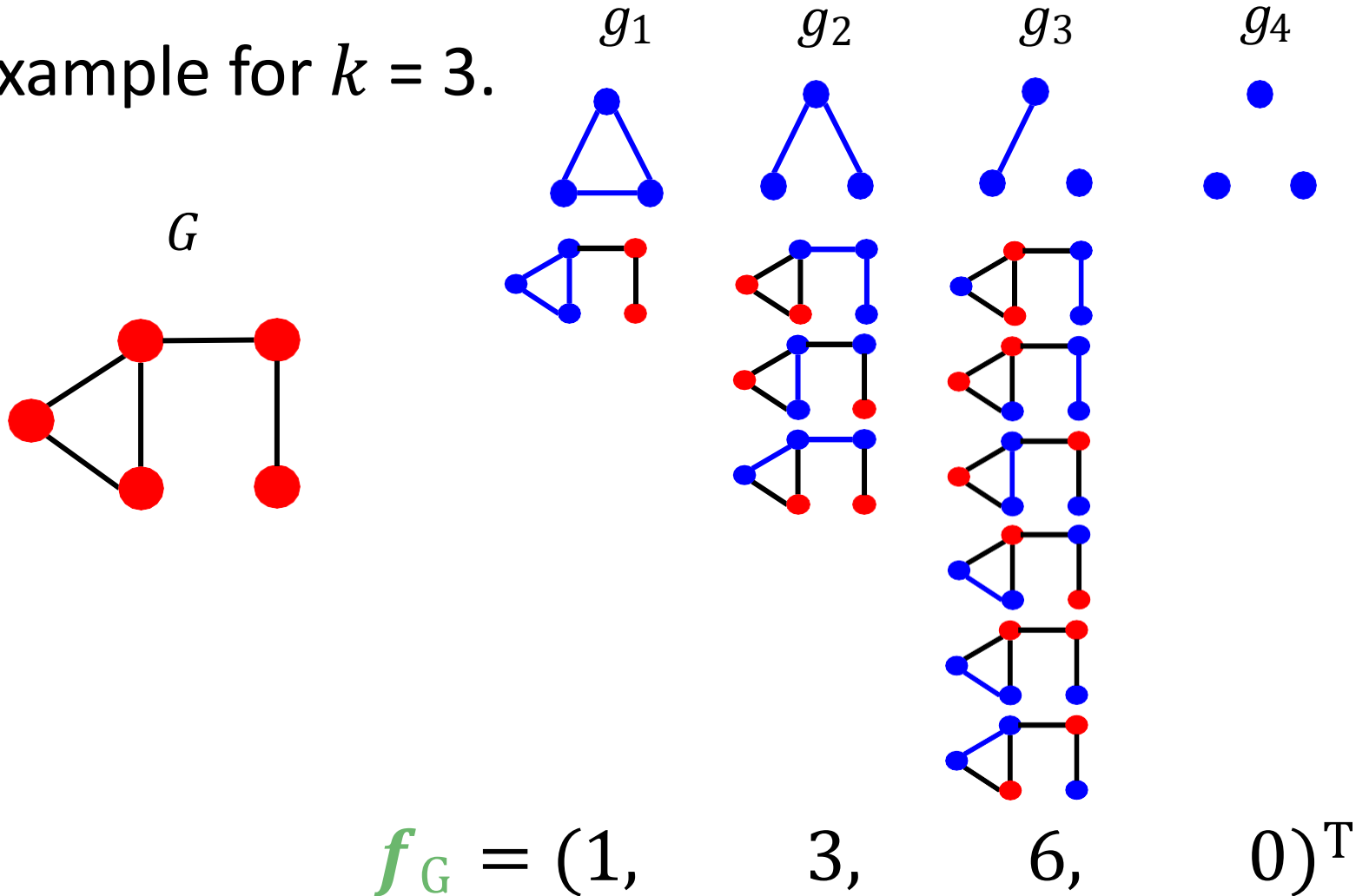
- For $k = 4$, there are 11 graphlets.



Shervashidze et al., AISTATS 2011

Graphlet Features

- Example for $k = 3$.



Graphlet Features

- Given graph G , and a graphlet list $\mathcal{G}_k = (g_1, g_2, \dots, g_{n_k})$, define the graphlet count vector $f_G \in \mathbb{R}^{n_k}$ as

$$(f_G)_i = \#(g_i \subseteq G) \text{ for } i = 1, 2, \dots, n_k.$$

Graphlet Kernel

- Given two graphs, G and G' , graphlet kernel is computed as

$$K(G, G') = \mathbf{f}_G^T \mathbf{f}_{G'}$$

- **Problem:** if G and G' have different sizes, that will greatly skew the value.
- **Solution:** normalize each feature vector

$$\mathbf{h}_G = \frac{\mathbf{f}_G}{\text{Sum}(\mathbf{f}_G)} \quad K(G, G') = \mathbf{h}_G^T \mathbf{h}_{G'}$$

Graphlet Kernel

Limitations: Counting graphlets is **expensive!**

- Counting size- k graphlets for a graph with size n
 - by enumeration takes n^k .
- This is unavoidable in the worst-case since **subgraph isomorphism test** (judging whether a
 - graph is a subgraph of another graph) is **NP-hard**.
- If a graph's node degree is bounded by d , an
 - $O(n d^{k-1})$ algorithm exists to count all the graphlets of size k .

Weisfeiler-lehman Kernel

- **Goal:** design an efficient graph featuredescriptor $\phi(G)$
- **Idea:** use neighborhood structure to iteratively "enrich" node vocabulary.
 - Generalized version of **Bag of node degrees** since node degrees are one-hop neighborhood information.

Color Refinement

- **Given:** A graph G with a set of nodes V .
 - Assign an initial color $c^{(0)}(v)$ to each node v .
 - Iteratively refine node colors by

$$c^{(k+1)}(v) = \text{HASH} \left(\left\{ c^{(k)}(v), \{c^{(k)}(u)\}_{u \in N(v)} \right\} \right),$$

where **HASH** maps different inputs to different colors.

- After K steps of color refinement, $c^{(K)}(v)$ summarizes the structure of K -hop neighborhood

Color Refinement

Also known as the 1-dimensional Weisfeiler-Lehman Algorithm

Algorithm

1: Multiset-label determination

- Assign a multiset-label $M_i(v)$ to each node v in G which consists of the multiset $\{l_{i-1}(u) | u \in \mathcal{N}(v)\}$.

2: Sorting each multiset

- Sort elements in $M_i(v)$ in ascending order and concatenate them into a string $s_i(v)$.
- Add $l_{i-1}(v)$ as a prefix to $s_i(v)$.

3: Label compression

- Map each string $s_i(v)$ to a compressed label using a hash function $f : \Sigma^* \rightarrow \Sigma$ such that $f(s_i(v)) = f(s_i(w))$ if and only if $s_i(v) = s_i(w)$.

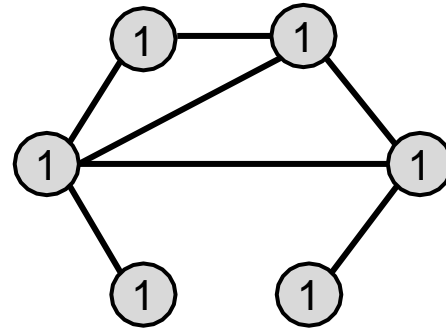
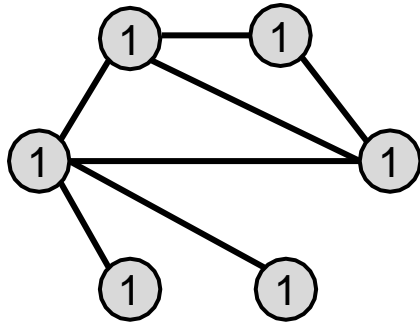
4: Relabeling

- Set $l_i(v) := f(s_i(v))$ for all nodes in G .
-

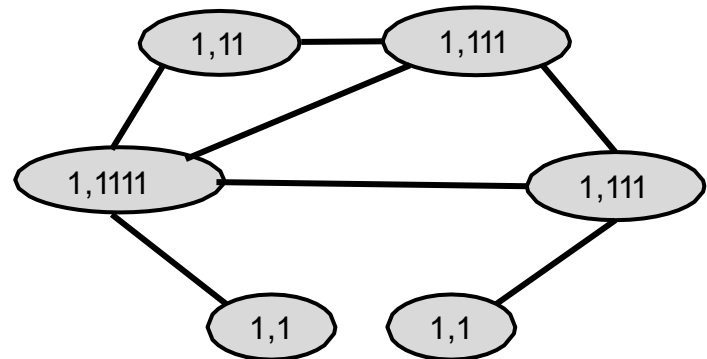
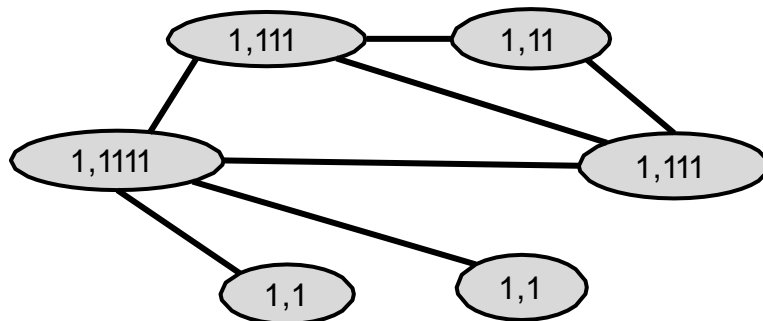
Color Refinement

Example of color refinement given two graphs

- Assign initial colors



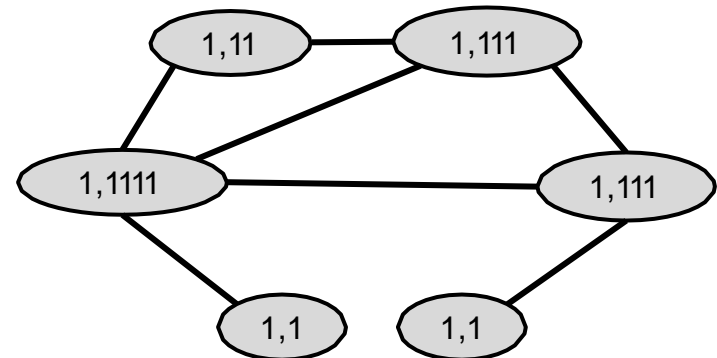
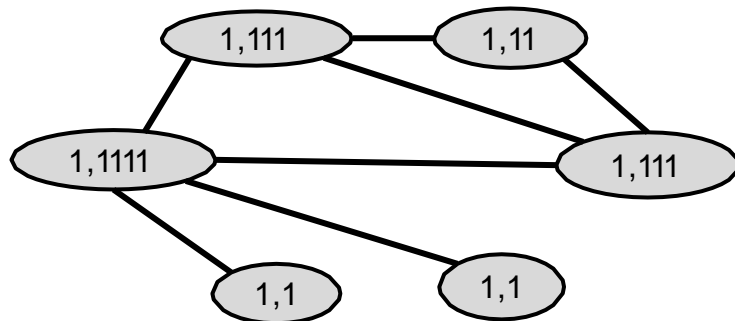
- Aggregate neighboring colors



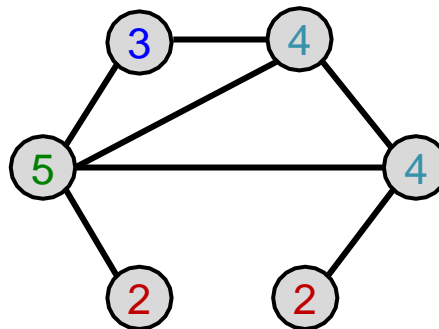
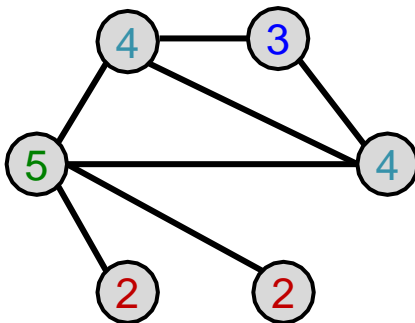
Color Refinement

Example of color refinement given two graphs

- Aggregated colors



- Hash aggregated colors



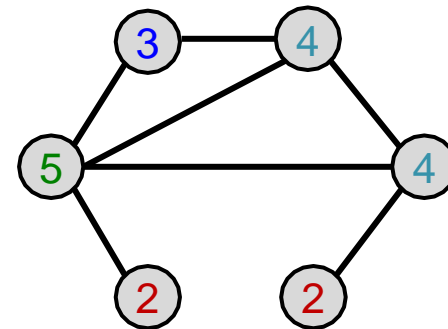
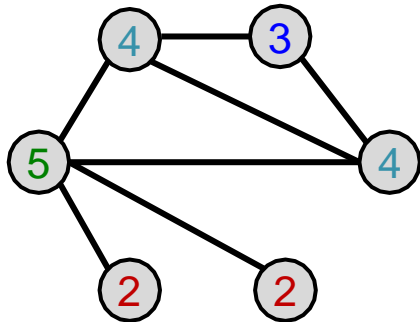
Hash table

1,1	-->	2
1,11	-->	3
1,111	-->	4
1,1111	-->	5

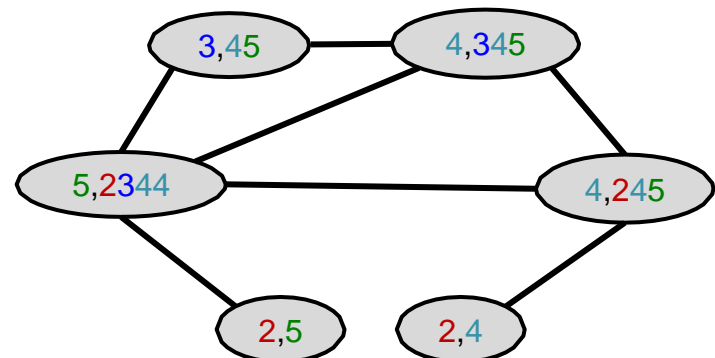
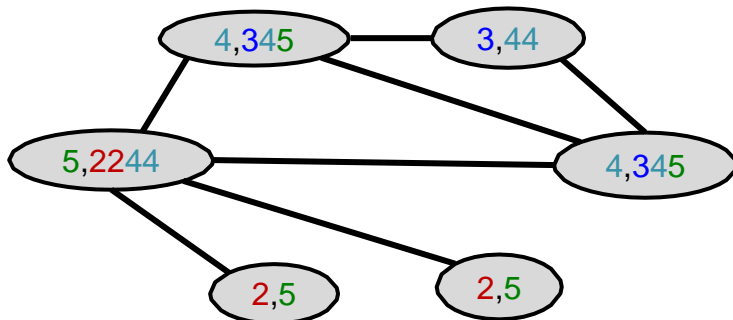
Color Refinement Algorithm (1)

Example of color refinement given two graphs

- Aggregated colors



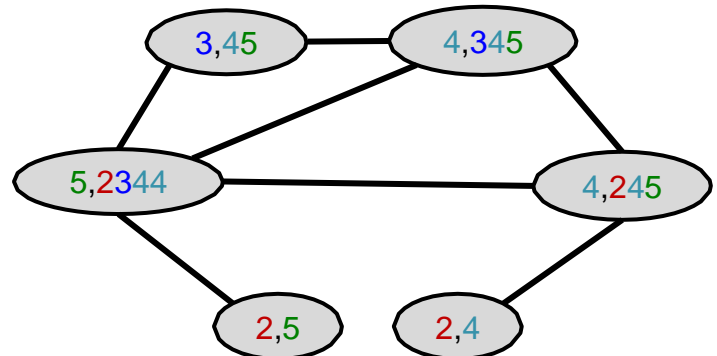
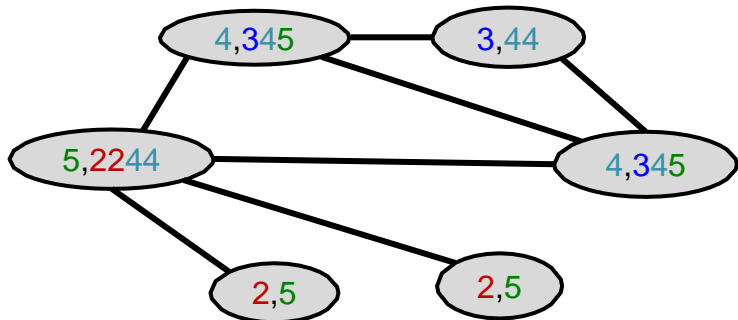
- Hash aggregated colors



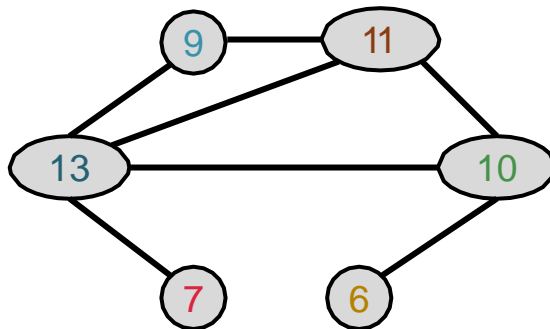
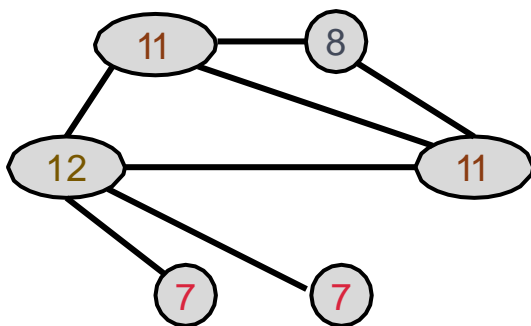
Color Refinement Algorithm (2)

Example of color refinement given two graphs

■ Aggregated colors



■ Hash aggregated colors



Hash table

2,4	-->	6
2,5	-->	7
3,44	-->	8
3,45	-->	9
4,245	-->	10
4,345	-->	11
5,2244	-->	12
5,2344	-->	13

Weisfeiler-lehman Features

After color refinement, WL kernel counts number of nodes with a given color.

$$\phi \left(\begin{array}{c} \text{Graph 1} \end{array} \right) = \begin{array}{c} \text{Colors} \\ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 \\ \text{Counts} \\ [6, 2, 1, 2, 1, 0, 2, 1, 0, 0, 2, 1, 0] \end{array}$$

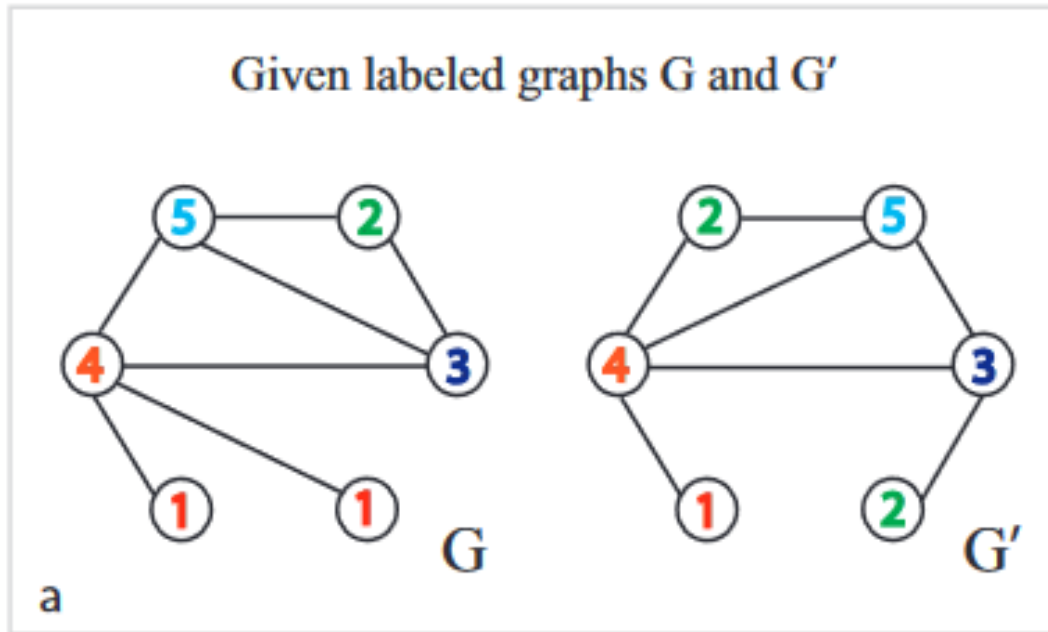
$$\phi \left(\begin{array}{c} \text{Graph 2} \end{array} \right) = \begin{array}{c} 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 \\ [6, 2, 1, 2, 1, 1, 1, 0, 1, 1, 1, 0, 1] \end{array}$$

Weisfeiler-Lehman Kernel

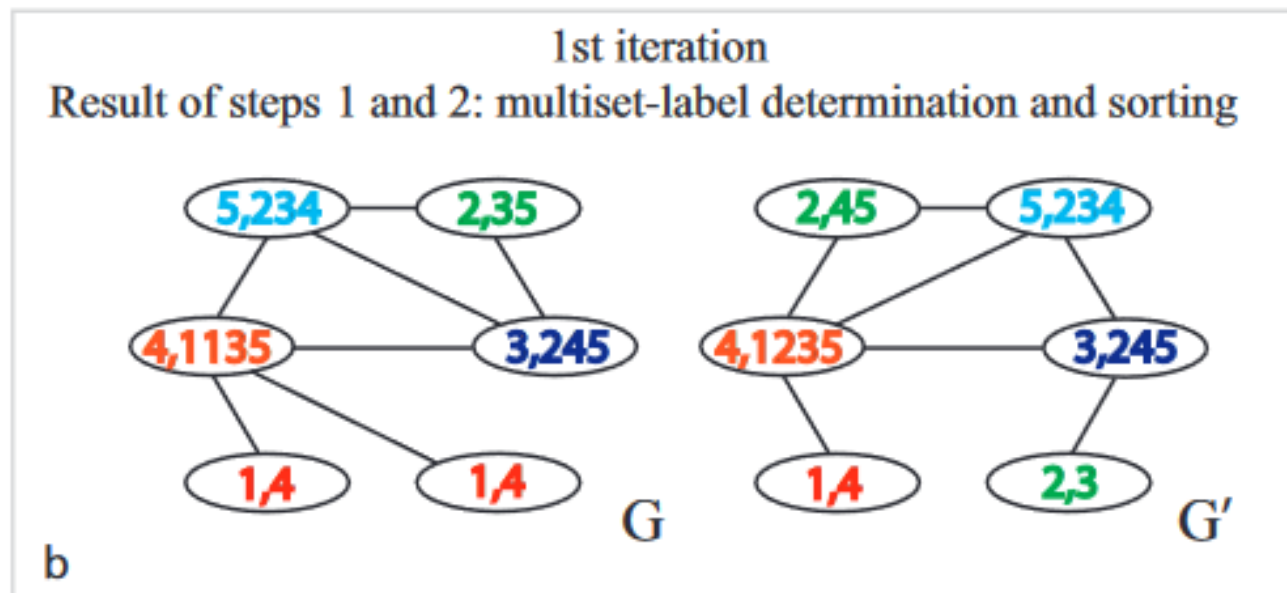
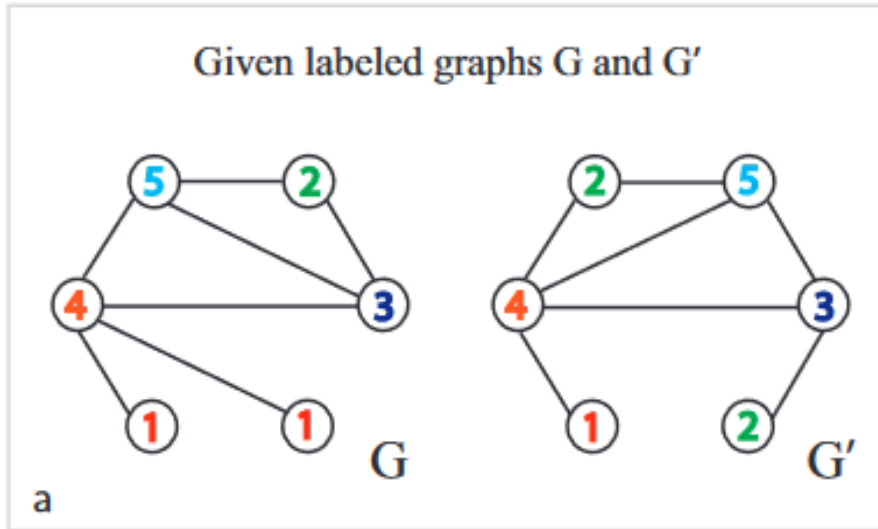
The WL kernel value is computed by the inner product of the color count vectors:

$$\begin{aligned} K(\text{graph}_1, \text{graph}_2) &= \phi(\text{graph}_1)^T \phi(\text{graph}_2) \\ &= 50 \end{aligned}$$

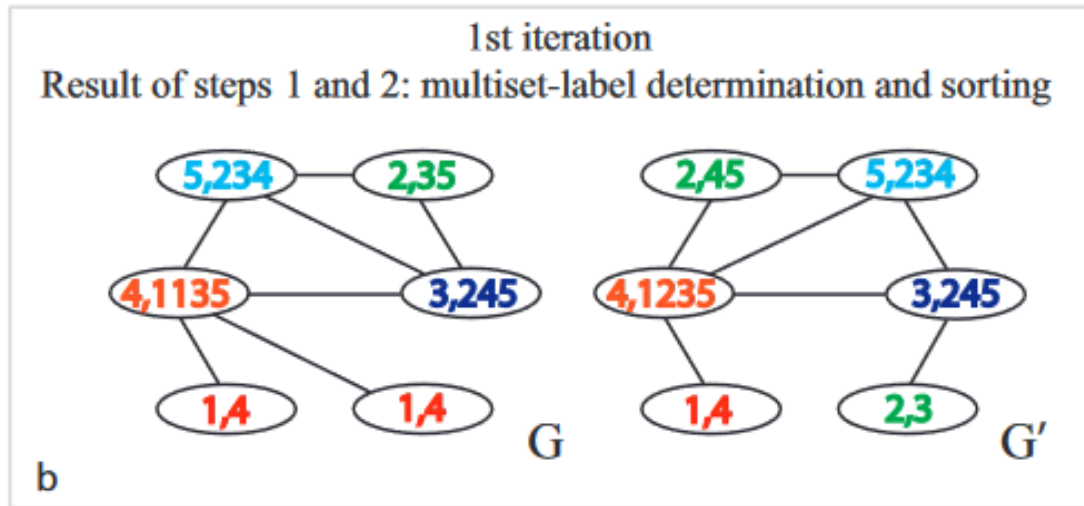
WL features on Labeled Graph (1)



WL features on Labeled Graph (2)



WL features on Labeled Graph (3)



1st iteration
Result of step 3: label compression

1,4	→	6	3,2,4,5	→	10
2,3	→	7	4,1,1,3,5	→	11
2,3,5	→	8	4,1,2,3,5	→	12
2,4,5	→	9	5,2,3,4	→	13

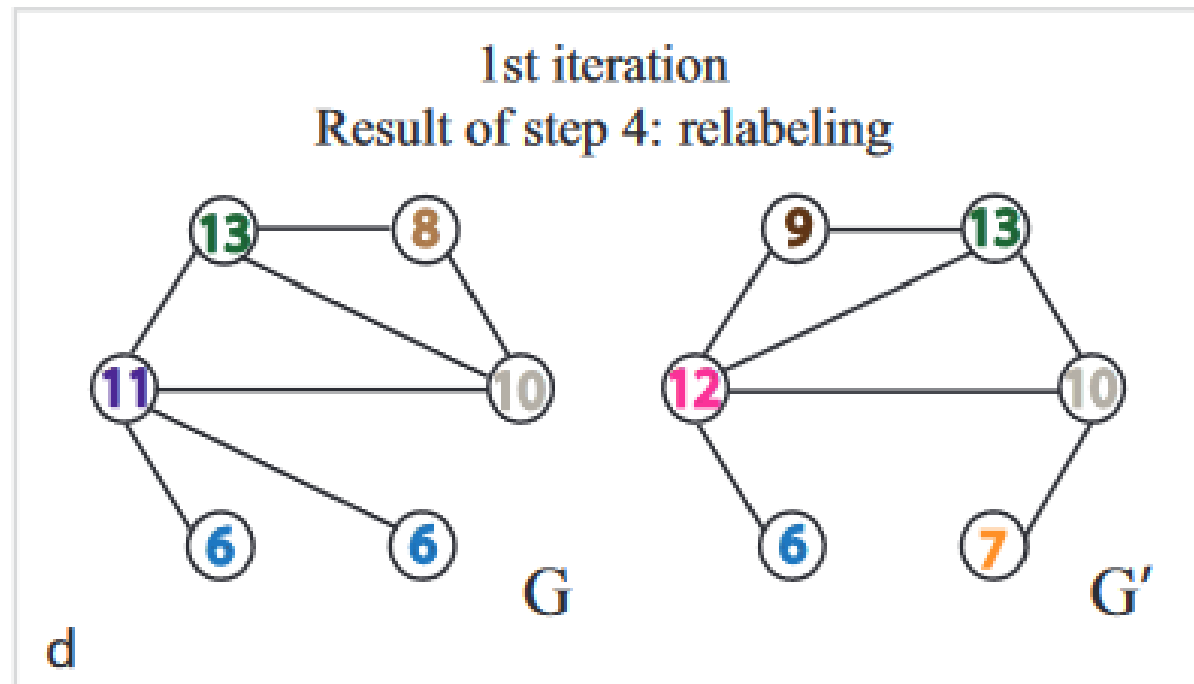
c

WL features on Labeled Graph (4)

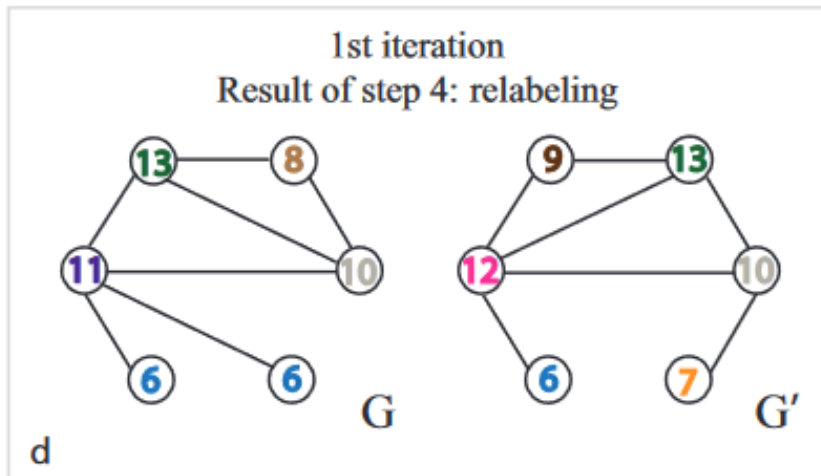
1st iteration
Result of step 3: label compression

1,4	→	6	3,245	→	10
2,3	→	7	4,1135	→	11
2,35	→	8	4,1235	→	12
2,45	→	9	5,234	→	13

c



WL features on Labeled Graph (5)



End of the 1st iteration
Feature vector representations of G and G'

$$\phi_{WLsubtree}^{(1)}(G) = (\mathbf{2}, \mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{2}, \mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{1})$$

$$\phi_{WLsubtree}^{(1)}(G') = (\mathbf{1}, \mathbf{2}, \mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{1})$$

<u>Counts of original node labels</u>	<u>Counts of compressed node labels</u>
---	---

$$k_{WLsubtree}^{(1)}(G, G') = \langle \phi_{WLsubtree}^{(1)}(G), \phi_{WLsubtree}^{(1)}(G') \rangle = 11.$$

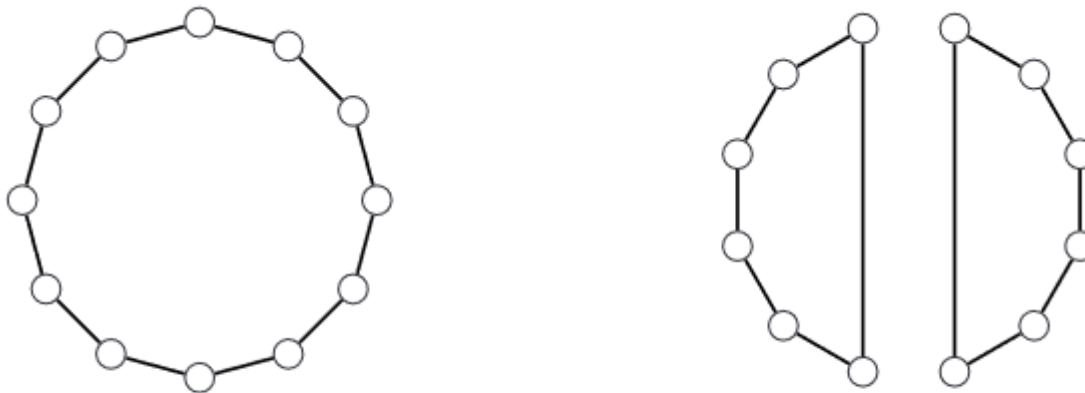
e

Weisfeiler-lehman Kernel

- WL kernel is **computationally efficient**
 - The time complexity for color refinement at each step is linear in $\#(\text{edges})$, since it involves aggregating neighboring colors.
- When computing a kernel value, only colors appeared in the two graphs need to be tracked.
 - Thus, $\#(\text{colors})$ is at most the total number of nodes.
- Counting colors takes linear-time w.r.t. $\#(\text{nodes})$.
- In total, time complexity is **linear in $\#(\text{edges})$** .

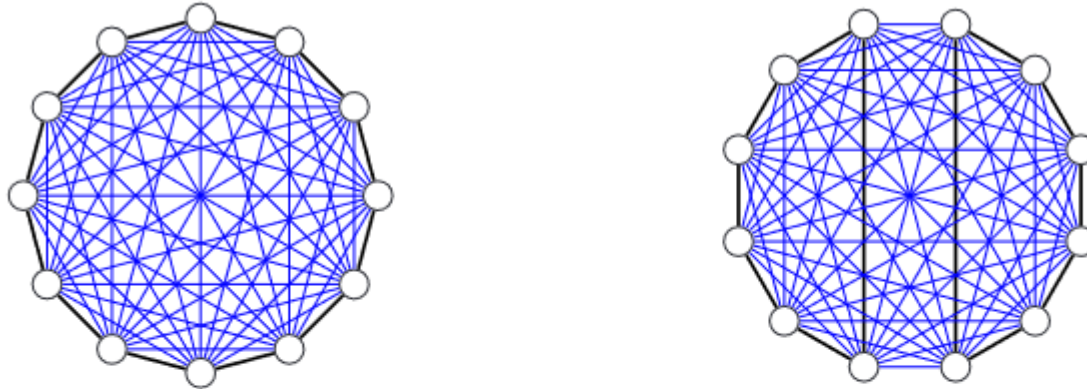
Example: a 2-dimensional Weisfeiler-Lehman Algorithm

- The k -dimensional Weisfeiler-Leman algorithm (k -WL), for $k \geq 2$, is a generalization of the 1-WL which colors tuples from $V(G)^k$ instead of nodes



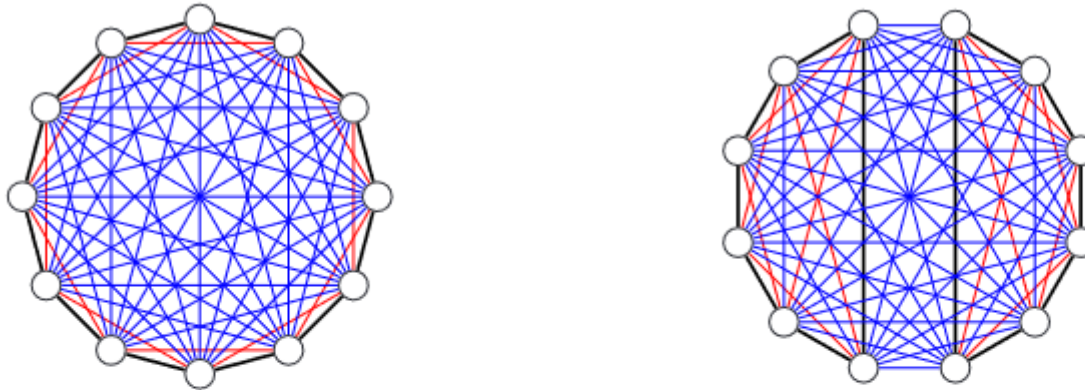
Input graphs: C_{12} and $2C_6$

Example: a 2-dimensional Weisfeiler-Lehman Algorithm



The initial coloring C^0 of V^2 : “edges” for (u, v) s.t. $u \sim v$,
“non-edges” for (u, v) s.t. $u \not\sim v$, “vertices” for (u, v) s.t. $u = v$.

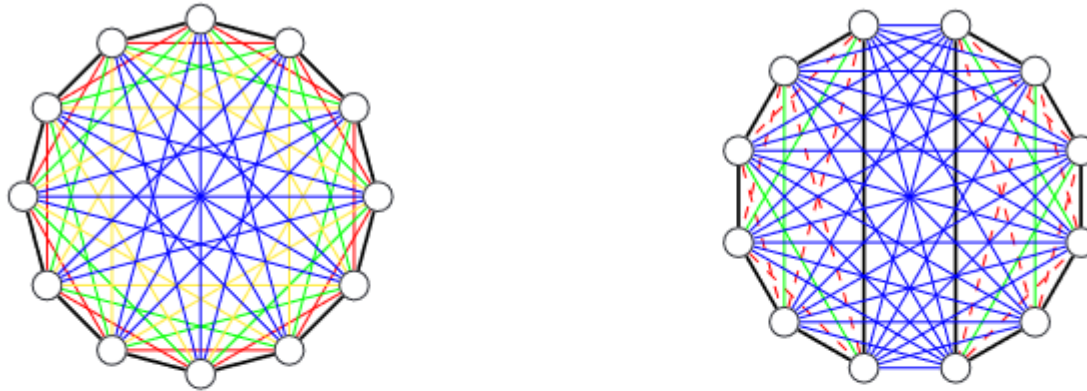
Example: a 2-dimensional Weisfeiler-Lehman Algorithm



A color refinement step:

$$C^{i+1}(u, v) = C^i(u, v) \mid \left\{ \left\{ C^i(u, w) \mid C^i(w, v) \right\} \right\}_{w \in V}$$

Example: a 2-dimensional Weisfeiler-Lehman Algorithm

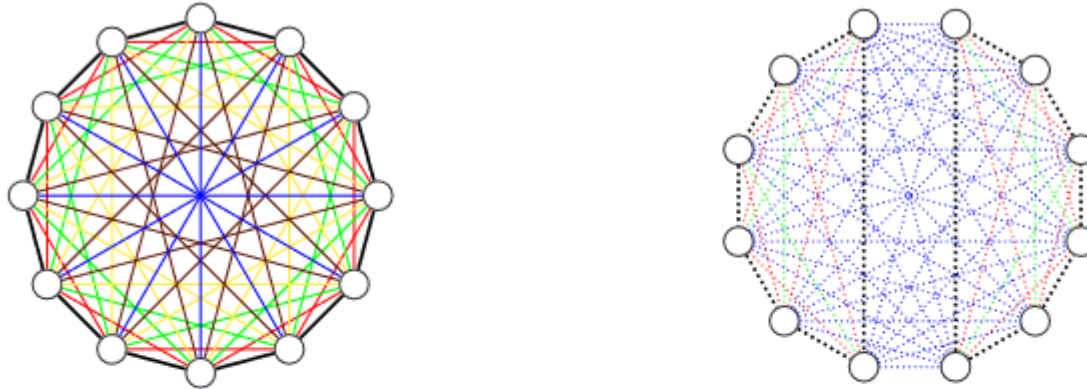


The next color refinement step:

$$C^2(u, v) = C^1(u, v) \mid \left\{ \left\{ C^1(u, w) \mid C^1(w, v) \right\}_{w \in V} \right\}$$

The non-isomorphism recognized.

Example: a 2-dimensional Weisfeiler-Lehman Algorithm



The next color refinement step:

$$C^3(u, v) = C^2(u, v) \mid \left\{ \left\{ C^2(u, w) \mid C^2(w, v) \right\} \right\}_{w \in V}$$

The color partition stabilized.

Graph-level Features: Summary

■ Graphlet Kernel

- Graph is represented as **Bag-of-graphlets**
- **Computationally expensive**

■ Weisfeiler-Lehman Kernel

- Apply K -step color refinement algorithm to enrich node colors
 - Different colors capture different K -hop neighborhood structures
- Graph is represented as **Bag-of-colors**
- **Computationally efficient**
- Closely related to Graph Neural Networks (as we will see!)

Learning Outcome

- **Traditional ML Pipeline**
 - Hand-crafted feature + ML model
- **Hand-crafted features for graph data**
 - **Node-level:**
 - Node degree, centrality, clustering coefficient, graphlets
 - **Link-level:**
 - Distance-based feature
 - local/global neighborhood overlap
 - **Graph-level:**
 - Graphlet kernel, WL kernel