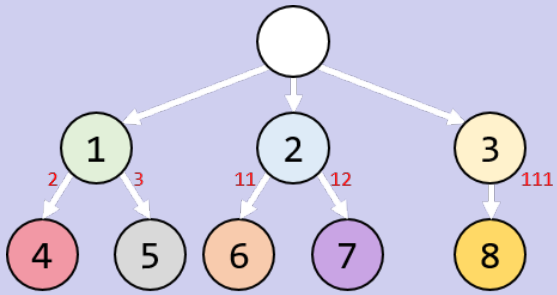


Master Thesis - Handout

"Learning Graph Similarities Using The Weisfeiler-Leman Label Hierarchy"

Fabrice Beaumont

Department of Information Systems and Artificial Intelligence
Dr. Pascal Welke

10. November 2022

Formal statement of the thesis project

In my thesis, I am going to
investigate the effect of learning similarities between Weisfeiler Leman labels and using these in the definition of a **graph similarity measure**, structured similarly to the Wasserstein Weisfeiler Leman graph kernel proposed in 2019 by Togninalli et al. [7].
The goal is to discuss, whether learning such similarities can improve the flexibility of the resulting graph similarity measure. The machine learning will be restricted in the sense that the ground distance is set as a a tree metric on the Weisfeiler Leman label hierarchy tree. The learning procedure includes iterative updates to the edge weights in this tree. Thereby changing the resulting distance between the Weisfeiler-Leman labels used for the similarity computation.

Definitions on graphs I

We define a **graph** as a tuple $G = (V, E)$, where V denotes a finite set of vertices and $E \subseteq 2^V$ a finite set of sets of edges on V. We also write $V(G)$ to refer to the vertex set of a graph G. In this thesis we consider edges to be undirected. That is an edge is a set of two distinct vertices:

$$E = \{\{v, w\} | v \neq w \ \wedge \ v, w \in V\} \tag{1}$$

Such graphs are sometimes called undirected simple graphs. Two graphs G_1 and G_2 are **isomorphic**, if there exists a bijective function between their vertices, that preserves all edges, and vertex labels. If for a graph $G = (V, E)$ and subsets $V' \subseteq V$ and $E' \subseteq E$ the tuple $T' = (V', E')$ is also a graph, we call it **subgraph** of G. We say a graph is **labeled**, if its vertices have categorical labels. These are denoted by a function

Definitions on graphs II

$L : V \to \Sigma$ for a finite alphabet Σ [7]. In the context of this thesis, we will have natural numbers as labels, such for an $s \in \mathbb{N}$ it is $\Sigma = [-1, s] \subseteq \mathbb{N}$. G is called attributed, if there exists a vertex attribute function $a : V \to \mathbb{R}$.
We say a graph is **weighted**, if there is a weight function $w : E \to \mathbb{R}$ on the edges of the graph. In the context of this thesis, we will have natural numbers as labels, such that there exists an $s \in \mathbb{N}$ such that $\Sigma = [-1, s]$.
When G is clear from the context, denote the number of vertices as $n := |V|$ and the number of edges as $m := |E|$.
Let G be a graph. We define a **walk** $\mathcal{W}(v_1, v_{k+1})$ in G from vertex $v_1 \in V$ to vertex $v_{k+1} \in V$ as a sequence of edges and vertices $\mathcal{W}(v1, v_{k+1}) = \{v_1, e_1, v_2, \ldots v_k, e_k, v_{k+1}\}$ where $e_i = \{v_i, v_{i+1}\} \in E$ for $i = 1, \ldots, k$. The **length of this walk** is denoted $|\mathcal{W}(v_1, v_{k+1})| = k$.
Given a weighted graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$, we

Definitions on graphs III

define the **distance** $d_G(v, w)$ between two vertices v and w in G as the sum of all weights of all edges in the $\mathcal{P}(v, w)$:

$$d_G(v, w) = \sum_{e \in \left(\mathcal{P}(v,w) \cap E\right)} w(e)$$

A graph G is called connected, if for every two vertices v and w there exists a walk $\mathcal{W}(v, w)$ in G. For a vertex $v \in V$ in a graph $G = (V, E)$ we define its **neighborhood** $\mathcal{N}(v)$ as the set of vertices (**neighbors**) n such that there exists an edge $e = \{v, n\} \in E$ between v and n (which is a path $\mathcal{P}(v, n)$ of length zero) [7]:

$$\mathcal{N}(v) = \{u \in V | \ \{u, v\} \in E\}$$

Definitions on graphs IV

More generally we define the *k*-th-order neighborhood $\mathcal{N}_k(v)$ of a vertex $v \in V$ in a graph $G = (V, E)$ as the set of vertices (k-th-order neighbors) n such that there exists a walk $\mathcal{W}(v, n)$ of length of at most k:

$$\mathcal{N}_k(v) = \{u \in V | \ |\mathcal{W}(v, k)| \leq k\}$$

Notice that $\mathcal{N}(v) = \mathcal{N}_1(v)$ and for $k > 1$, $\mathcal{N}(v) \neq \emptyset$ implies $v \in \mathcal{N}_k(v)$. The degree $\delta(v)$ of a vertex $v \in V$ is given as the size of its first neighborhood, that is $\delta(v) := |\mathcal{N}(v)|$.
We define a path $\mathcal{P}(v_1, v_{k+1})$ as a walk $\mathcal{W}(v_1, v_{k+1})$ such that no vertex or edge in the walk equals another ($e_i \neq e_j$ for all $1 \leq i < j \leq k$ and $v_i \neq v_j$ for all $1 \leq i < j \leq k + 1$). We define a circle as a walk $\mathcal{W}(v_1, v_{k+1})$ such $v_1 = v_{k+1}$ and no other vertex and no edge equals another. We define a tree as a connected graph $T = (V, E)$, which does

Definitions on graphs V

not contain any circles [2]. Notice that this implies that all paths in T are unique. That is there cannot be two different path between two vertices. We extend the notion of subgraphs to subtrees. A tree $T = (V, E)$ is called **rooted** if there exists a distinguished vertex $r \in V$ called **root**. In every tree we call vertices with only one adjacent edge (that is a vertex v such that $|\{e \in E : \ v \in e\}| = 1$) the **leaves** of the tree. The **depth** of a vertex $v \in V$ in a rooted tree with root r is defined as length of the path between v and r ($\mathcal{P}(v, r)$). We say the root has length zero. For two neighboring vertices v, w and the edge that connects them $e = \{v, w\}$ we call v the **parent** of w (v_p), if v has smaller depth than w. w is then called **child** of v (v_c). Notice that in this definition w cannot be the root r, since no vertex has smaller depth than r. Thus the root has no parent and we set $r_p = \emptyset$. We extend the definition to the connecting edge e and

Definitions on graphs VI

say v is the parent vertex of e ($e_p = w$) and w is the child vertex of e ($e_c = w$). All vertices on the path from v to the root r are called **ancestors** of v. We call a tree **leveled**, if all paths between all leaves and the root have the same length.

Definitions on tree metrics I

A metric $d : \Omega \times \Omega \to \mathbb{R}$ is called a **tree metric** on Ω if there exists a tree $T = \{V, E\}$ with non-negative edge weights such that all elements of Ω are contained in its vertices ($\Omega \subseteq V$) and such that for every $v, w \in \Omega$, one has that $d(v, w) = d_G(v, w)$ equals to the distance between the vertices v and w in T [4].In this work we will use tree metrics with $\Omega = V$. Also note that we will only consider rooted trees and compute distances between vertices that have equal depth. This greatly reduces the actually used domain for our tree metric.

Definitions on the used Wasserstein Distance I

Let $T = (V, E)$ be a rooted, weighted tree with edge weights w, a root $r \in V$ and a tree metric d_T. We define $\Gamma(x)$ as the set of vertices in the rooted subtree of T that contains x as root and all other vertices have higher depth than x in T. That is to say all vertices, such that the unique path $\mathcal{P}(r, z)$ from them to the root r contains the vertex x:

$$\Gamma(x) := \{z \in V(T) |\ x \in \mathcal{P}(r, z)\}$$

Using this, we can define the **closed form, negative definite Tree Wasserstein Distance** on T in the following way [4].For two Borel probability distributions μ, ν on V, the Tree Wasserstein Distance can be computed as

$$\mathbb{W}_{d_T}(\mu, \nu) := \sum_{e \in E} w(e) \left| \mu\big(\Gamma(e_c)\big) - \nu\big(\Gamma(e_c)\big) \right|$$

Definitions on Weisfeiler-Leman labels I

The **Weisfeiler-Leman** (WL) **labeling scheme** is vertex labeling scheme at the center of the Weisfeiler-Leman test of graph isomorphism proposed by Boris Weisfeiler and Andrei Leman [8]. In this thesis we will consider the 1-dimensional version of the WL labeling scheme. It is also denoted as WL labeling scheme, 1-WL vertex embedding, WL color refinement, naive vertex refinement. In terms of the higher-dimensional k-WL labeling scheme it is equivalent to 2-WL labeling scheme and thus the 2-variable language with counting [6, 9, 1].
The scheme iteratively propagates neighborhood information by compressing it in vertex labels [8]. These vertex labels will be called WL-labels, where the zeroth WL-labels can also denote original vertex labels.

Definitions on Weisfeiler-Leman labels II

Algorithm 1 WL labeling scheme

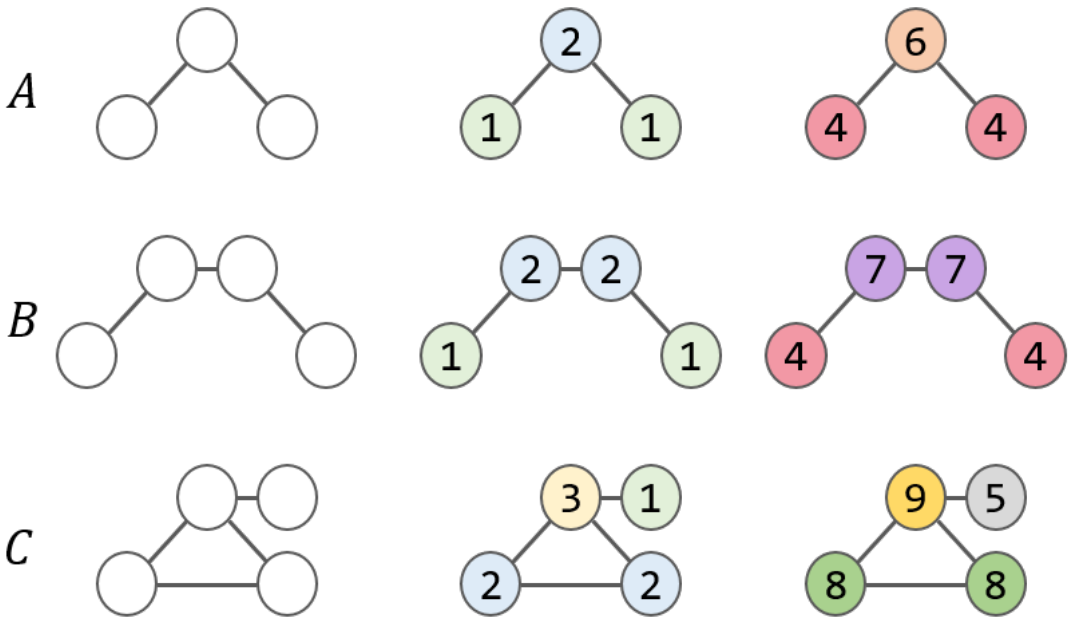
Input: a graph $G = (V, E, L)$ and
a perfect hash function.
Output: a vector representation of G of size $|V|$.

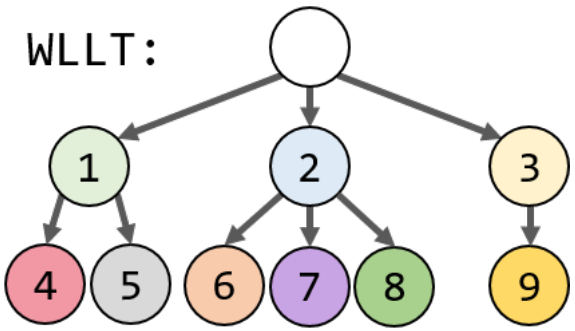
1: $\ell^0(v) \leftarrow \text{hash}(L(v))$ for all $v \in V$
2: $\left(\ell^1(v)\right)_{v \in V} = \emptyset$
3: $k = 1$
4: **while** $\left(\ell^k(v)\right)_{v \in V} \neq \left(\ell^{k-1}(v)\right)_{v \in V}$ **do**
5: $\quad \ell^k(v) \leftarrow \text{hash}\left(\ell^{k-1}(v), \{\!\{\ell^{k-1}(w) |\ w \in \mathcal{N}_G(v)\}\!\}\right)$ for all $v \in V$
6: $\quad k = k + 1$
7: **end while**
8: **return** $\left(\ell^{k-1}(v)\right)_{v \in V}$

Definitions on Weisfeiler-Leman labels III

The procedure was generalized by [9] with $\ell^0(v) = L(v)$ for labeled graphs and $\ell^0(v) = |\mathcal{N}(v)|$ for unlabeled graphs [5]. However using uniform labels $\ell^0(v) = 0$ for unlabeled graphs results in an equivalent labeling, just delayed by one iteration.
To obtain a maximally powerful embedding, the aggregation and hashing step (line 5) must be injective [9]. This injective aggregation is often realized by sorting the WL-labels of the neighborhood first. For the hash function we will consider the set of integers as its range. More precisely, the hash function shall map the given inputs to the next free integer, that is not already the image to an input. Notice how WL-labels (of deeper iterations) relate iteratively to the n-th order neighborhoods of the vertices. Thus comparing two WL-labels relates to comparing complete neighborhoods, similarly to subgraphs. More accurately, WL-labels are equivalent to so called unfolding trees.

Weisfeiler Leman labeling scheme





Weisfeiler Leman labeling tree I

Given k iterations of WL labelings on a database of graphs $\mathcal{G}$, one can construct a WLLT T. Labeled and un-labeled graphs can be treated in the same way, by considering vertex labels as the zeroth WL-labels. With this in mind, use an artificial label to initialize the root of T. The k-th layer of T consists of all arising WL labels in iteration k of the labeling in any graph . That is, there exists an edge (l, l') in the tree between vertices l and l' if there exists a vertex $v \in V(G)$ in some graph $G \in \mathcal{G}$ and $k \in \mathbb{N}$ with $l = \ell^k(v)$ and $l' = \ell^{k+1}(v)$. The complete construction of the WLLT is sketched in algorithm 2.

Weisfeiler Leman labeling tree II

The k-th WL label of vertex v is given by $\ell_k(u)$. Notice that if two vertices have different WL labels in one refinement step, this implies that they have different WL labels in all later refinement steps [8, 3]:

$$\forall u, v \in V \; \forall i \in \mathbb{N}: \quad \ell_k(u) \neq \ell_k(v) \implies \ell_{k+1}(u) \neq \ell_{k+1}(v)$$

Hence the sequence of WL labels gives rise to a hierarchy which can be represented in a tree.

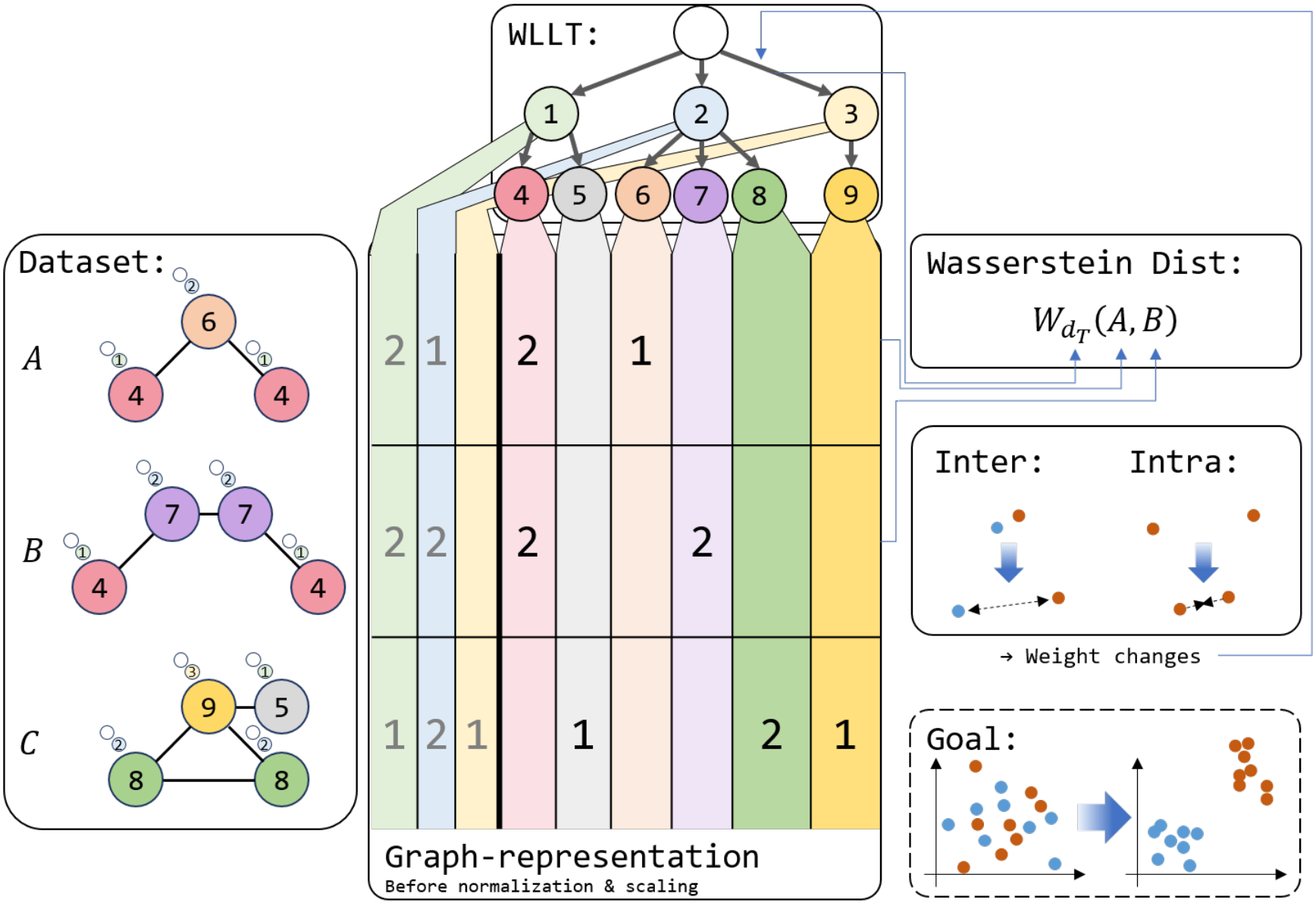
Weisfeiler Leman labeling tree III

Algorithm 2 WLLT construction

Input: a graph dataset $\mathcal{G}$,
WL labeling depth k.
Output: WLLT T of height k (unweighted).

Initialize the WLLT as a single root $T = (r, \emptyset)$
for G in $\mathcal{G}$ **do**
    for $i = 0$ to k **do**
        Generate the i-th WL-feature F of G
        for $l \in F$ **do**
            if $l \notin T$ **then**
                $V(T) = V(T) \cup \{l\}$
                $E(T) = E(T) \cup (p, l)$ where $c^{i-1}(v) = p$ and $c^{i}(v) = l$
            end if

Method sketch



- J.-Y. Cai, M. Furer, and N. Immerman. "An optimal lower bound on the number of variables for graph identification". In: *30th Annual Symposium on Foundations of Computer Science*. IEEE, 1992. DOI: 10.1109/sfcs.1989.63543.
- Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer Berlin Heidelberg, 2018. DOI: 10.1007/978-3-662-56039-6.
- Nils M. Kriege et al. "On Valid Optimal Assignment Kernels and Applications to Graph Classification". In: 2016.
- Tam Le et al. "Tree-Sliced Variants of Wasserstein Distances". In: *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. 2019.


Nino Shervashidze and Karsten M. Borgwardt. "Fast subtree kernels on graphs". In: *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*. Ed. by Yoshua Bengio et al. Curran Associates, Inc., 2009, pp. 1660–1668.


Nino Shervashidze et al. "Weisfeiler-Lehman Graph and Kernels". In: *J. Mach. Learn. Res.* 12 (2011), pp. 2539–2561.


Matteo Togninalli et al. "Wasserstein Weisfeiler-Lehman Graph Kernels". In: *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. 2019.


Weisfeiler and Leman. "THE REDUCTION OF A GRAPH TO CANONICAL FORM AND THEALGEBRA WHICH APPEARS THEREIN". In: *Journal of Applied Mathematics and Physics*. 1968.

 Keyulu Xu et al. “HOW POWERFULARE GRAPH NEURAL NETWORKS?” In: *International Conference on Learning Representations*. 2019.