Supplementary materials for the article

Network alignment and similarity reveal atlas-based topological differences in structural connectomes

Matteo Frigo[1,3], Emilio Cruciani[2,3], David Coudert[2], Rachid Deriche[1], Emanuele Natale[2], and Samuel Deslauriers-Gauthier[1]

[1]Université Côte d'Azur, Inria, France
{matteo.frigo,rachid.deriche,samuel.deslauriers-gauthier}@inria.fr
[2]Université Côte d'Azur, Inria, CNRS, I3S, France
{emilio.cruciani,david.coudert,emanuele.natale}@inria.fr
[3]These authors contributed equally to this work.

A The generalized Weisfeiler-Lehman alignment algorithm

In this section we describe our alignment algorithm, WL-align, made of two parts:

- A generalized version of the famous Weisfeiler-Lehman (non-)isomorphism test, WL-sign, described in Section A.1, which computes node *signatures* (an embedding of nodes into $\mathbb{R}^m$ for some m);
- An optimal assignment problem associated to the pairwise distances among signatures, described in Section A.2.

We first recall the classical Weisfeiler-Lehman algorithm (WL) (Färer, 2017). At the onset, we color nodes of $G = (V, E)$ with the same color. We then refine the coloring in rounds: at round $i + 1$, nodes u and v receive new and different colors whenever they already had different colors at round i, or if the multi-sets of colors of their neighbors at round i were different. For example, if two nodes have different degrees, then the multi-set of neighbor colors are initially different (i.e., their respective cardinality differ), and they will be assigned a different color after the first iteration of the algorithm.

The colors assigned by the algorithm at each round define a partition of the nodes, which stabilizes after at most n rounds. Since WL is solely based on the topology of the graph, if the partitions that it constructs for two graphs G_1 and G_2 differ then the two graphs are not isomorphic. If, instead, the two partitions are the same, the generated partitions can be exploited to search for an isomorphism between the two graphs (McKay and Piperno, 2014).

A.1 Extension of WL to weighted graphs

We now describe WL-sign, an algorithm that computes node signatures for weighted graphs. After describing the algorithm, we provide some intuition on how it relates to the original WL algorithm.

For each node u of a graph $G = (V, E, w)$ with positive weights, WL-sign computes a vector $H_u \in \mathbb{R}^{\Delta}$ which we call *signature* of u, where $\Delta := \sum_{i=0}^{\ell} k^i$ and where k, ℓ are input parameters of the algorithms which limit the number of explored paths for efficiency reasons. In particular, the quantity Δ corresponds to the number of nodes in a complete k-ary tree of depth ℓ.

Given a path $\pi = (v_0, \dots, v_h)$, we define $\mathrm{f}(\pi)$ as

$$\mathrm{f}(v_0,\ldots,v_h) := \begin{cases} \mathrm{vol}(v_0) & \text{if } \pi = (v_0), \\ \frac{w(v_{h-1},v_h)}{\mathrm{vol}(v_{h-1})} \cdot \mathrm{f}(v_0,\ldots,v_{h-1}) & \text{otherwise.} \end{cases} \quad (1)$$

The algorithm to compute the signature H_u of a given node u is the following.

1. Initially append the zero-length, i.e., single-node, path $\pi = (u)$ to a queue Q (FIFO data structure).
2. Until the queue Q is empty, perform the following actions:
   - Pop[1] a path $\pi = (u,\ldots,v_h)$ from the queue Q.
   - Set the next entry of the vector H_u equal to $\mathrm{f}(\pi)$.
   - If $h < \ell$, i.e., the length of the path π is strictly less than ℓ, then:
     - Let $z_1,\ldots,z_n$ be an ordering of the nodes such that $w(v_h,z_1) \geq \ldots \geq w(v_h,z_n)$, with ties broken uniformly at random, and let $\pi_z = (u,\ldots,v_h,z)$.
     - Add to the queue Q the k paths $\pi_{z_1},\ldots,\pi_{z_k}$.
3. Return H_u.

A.1.1 Insights on the design of WL-sign

The intuition behind the above procedure is the following.

By iteratively coloring nodes with different colors, based on the colors present in their neighborhood, the WL algorithm can be seen as implicitly performing a BFS search with repetition (i.e., which revisits already-visited nodes), while keeping track of the degree sequence encountered at each level of such BFS. WL-sign performs such BFS, while limiting it according to the parameters ℓ and k, for efficiency reasons.

Note also that the signatures produced by the WL algorithm have a binary meaning: nodes with different signatures cannot be mapped into one another by an isomorphism. The signatures produced by the WL-sign algorithm, instead, are meant to be used for estimating the similarity of nodes in order to construct a network alignment (see Appendix A.2). Hence, in order to prevent nodes which are further from u to influence the similarity score more than nodes closer to it, in Appendix 1 the volume of node v at distance i gets multiplied by some normalizing factor; the latter can be regarded as the probability of terminating on v when performing i steps of a random walk starting from u. As a motivating example for such normalizing factor, we can see that, in the special case of an unweighted and almost regular graph, the normalizing factor is ensuring that all levels of the aforementioned BFS contributes comparably to the signature.

A.2 From the signatures to the alignment

Let $\{H_u^{(1)}\}_{u\in V_1}$ and $\{H_u^{(2)}\}_{u\in V_2}$ be the sets of signatures constructed by the WL-sign algorithm for the nodes of two graphs G_1 and G_2, respectively, and let $\mathrm{dist}(H_u^{(1)}, H_u^{(2)})$ denote the euclidean distance between two signatures. Our experiments show that the distances among signatures enable to efficiently construct a good network alignment by employing the following natural idea.

The pairwise distances among signatures can be regarded as edges of a weighted bipartite graph whose nodes are the nodes of the two graphs, that is $G_{1,2} = ((V_1,V_2), V_1 \times V_2, w)$ where $w(u,v) = \mathrm{dist}(H_u^{(1)}, H_v^{(2)})$. We can then consider the alignment given by the minimum weight matching of $G_{1,2}$. Such matching can be efficiently computed via classical algorithms for the optimal assignment problem, such as the Hungarian algorithm (Kuhn, 1955). The procedure resulting from combining WL-sign with the aforementioned use of the Hungarian algorithm constitute our final algorithm WL-align.

[1] That is, get and remove from the queue.

B Correction of FAQ

The sign of the gradient of f is inconsistent in (Vogelstein et al., 2015, Section 3). Moreover Equations (9) and (10) should be convex combinations (i.e., $P^{(i)}$ should be multiplied by $(1 - \alpha)$ in Steps 3 and 4) in order to obtain a stochastic matrix.

References

M. Färer. On the Combinatorial Power of the Weisfeiler-Lehman Algorithm. In D. Fotakis, A. Pagourtzis, and V. T. Paschos, editors, *Algorithms and Complexity*, Lecture Notes in Computer Science, pages 260–271, Cham, 2017. Springer International Publishing. ISBN 978-3-319-57586-5. doi: 10.1007/978-3-319-57586-5_22.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2): 83–97, 1955. doi: 10.1002/nav.3800020109.

B. D. McKay and A. Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, Jan. 2014. ISSN 0747-7171. doi: 10.1016/j.jsc.2013.09.003.

J. T. Vogelstein, J. M. Conroy, V. Lyzinski, L. J. Podrazik, S. G. Kratzer, E. T. Harley, D. E. Fishkind, R. J. Vogelstein, and C. E. Priebe. Fast Approximate Quadratic Programming for Graph Matching. *PLoS ONE*, 10(4), Apr. 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0121002.