Graph Neural Networks and Optimal Transport for Distance Learning between Graphs

LARGR Workshop Lille 2023

Aldo Moscatelli[1,2]
Tutors : Sébastien Adam[2],
Maxime Berar[2], Pierre Héroux[2].

[1] Presentation author, aldo.moscatelli@univ-rouen.fr
[2] LITIS Lab, University of Rouen Normandy

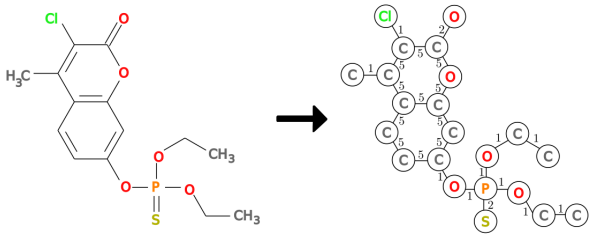
March 10, 2023

ED 590 MIIS
Mathématiques, Information, Ingénierie des Systèmes

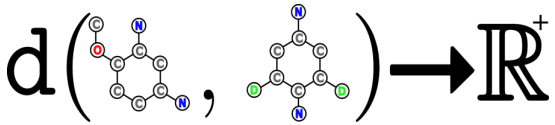
UNIVERSITÉ DE ROUEN NORMANDIE

litis

Graph representation



Challenge: graphs are non-euclidean objects and scalar products are not defined for graphs.
How to define the distance between two graphs?

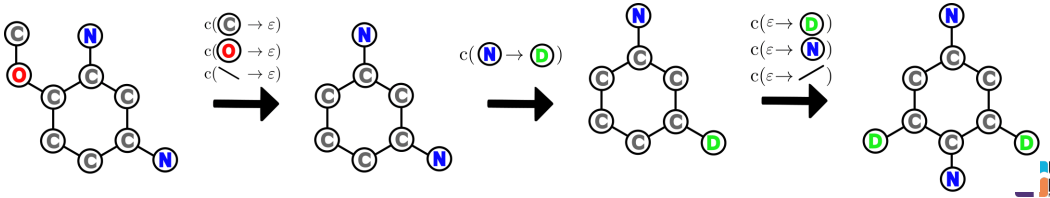


Graph Edit Distance (GED)

$$GED(G_1, G_2) = \min_{e_1,\ldots,e_k \in P(G_1,G_2)} \sum_{i=1}^{k} c(e_i)$$

with (e_i) edit operations of the path from G_1 to G_2.

Set of edit operations
- **node insertion**
- **node deletion**
- **node substitution**
- **edge insertion**
- **edge deletion**
- **edge substitution**



Traditional approaches

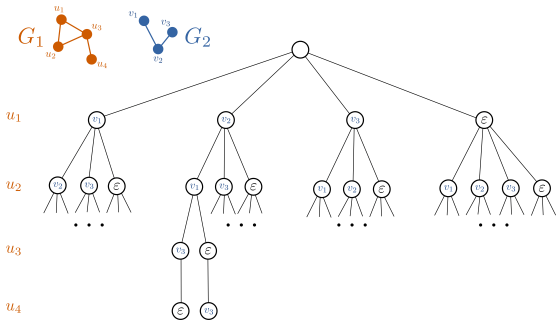
Exact computation :

- $\mathbf{A}^*$ algorithm with heuristic
- Depth-first search algorithm
- QAP solve with a constrained ILP solver (F1, F2)

NP-Hard problem

Approximation methods:
node-based methods

- LSAP relaxation of QAP
- Bipartite matching (BPGED)
- Hausdorff matching



Can we learn it?

Why?

- To by-pass the complexity
- To have a better approximation

How?

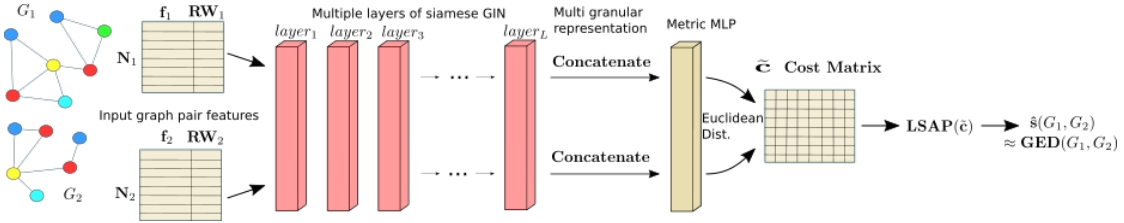
- Siamese architecture for graph pairs
- Through Graph Neural Network(GNN) for nodes embedding
- Optimal transport/LSAP to match sets of nodes embedding and infer distance cost

Already existing Deep Learning methods:

- SimGNN/GotSim/Greed...

UNIVERSITÉ DE ROUEN NORMANDIE

Our architecture Presentation



Nodes embedding with GNN

We use Graph Isomorphism Network (GIN) to learn an embedding of the nodes of the graph by aggregating each node's neighborhood at each layer:

$$\mathbf{h}_v^{(l)} = MLP^{(l)}\left(\left(1+\epsilon^{(l)}\right).\mathbf{h}_v^{(l-1)} + \sum_{u\in\mathcal{N}(v)} \mathbf{h}_u^{(l-1)}\right) \tag{1}$$

Where l is the l-th layer of GIN, $\mathcal{N}(v)$ the one-hop neighborhood of node v, $\epsilon^{(l)}$ a learnable parameter and MLP a multi-layer perceptron with two layers.
We then obtain, for L layers of GIN, the final nodes embedding :

$$\mathbf{H}_v = MLP^{(metric)}\left(\mathbf{h}_v^{(1)}|\mathbf{h}_v^{(2)}|...|\mathbf{h}_v^{(L-1)}|\mathbf{h}_v^{(L)}\right) \tag{2}$$

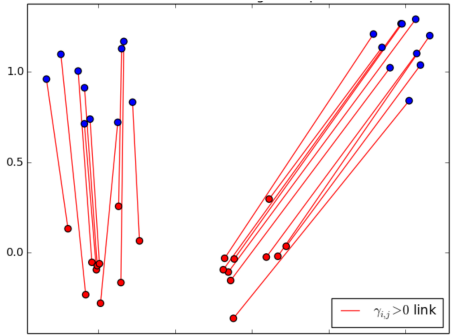
UNIVERSITÉ DE ROUEN NORMANDIE

Discrete Optimal Transport

We will formulate this problem as an instance of Monge's Problem equivalent to the Linear Sum Assignment Problem (LSAP).
By introducing permutation matrix $\mathbf{X} = (x_{i,j})$ we can define LSAP as follow :

$$\textbf{LSAP}(\tilde{\mathbf{C}}) = \min_{\mathbf{X}} \sum_{i=1}^{N} \sum_{j=1}^{N} \tilde{c}_{i,j} x_{i,j}$$

$$\tilde{\mathbf{C}} = \left(\begin{array}{ccc|ccc} \mathbf{c}_{1,1} & \cdots & \mathbf{c}_{1,N_2} & \mathbf{d}_1 & \cdots & \infty \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}_{N_1,1} & \cdots & \mathbf{c}_{N_1,N_2} & \infty & \cdots & \mathbf{d}_{N_1} \\ \hline \mathbf{a}_1 & \cdots & \infty & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \infty & \cdots & \mathbf{a}_{N_2} & 0 & \cdots & 0 \end{array}\right)$$



Substitution costs

Let be a graph pair (G, G') of size (N_1, N_2). We obtain as defined in (2), two sets of embedded nodes $\mathbf{H}, \mathbf{H}'$.
We can now compute a cost matrix by using the Euclidean distance between nodes representation.

$$\mathbf{C}_{i,j} = \left\| \mathbf{H}_i - \mathbf{H}'_j \right\|_2 \qquad \mathbf{C} = \begin{pmatrix} \mathbf{c}_{1,1} & \cdots & \mathbf{c}_{1,N_2} \\ \vdots & \ddots & \vdots \\ \mathbf{c}_{N_1,1} & \cdots & \mathbf{c}_{N_1,N_2} \end{pmatrix}$$

Costs of $\mathbf{C}$ correspond to GED's substitutions cost of nodes of G with those of G'.

Insertion and deletion costs

We can also use the Euclidean distance to define insertion and deletion costs. In this case, we use the norm of the embedding vector, it represents the distance to 0.

$$\text{for deletion : } \mathbf{D}_{i,j} = \begin{cases} \|\mathbf{H}_i\|_2 \text{ if } i = j, \text{ with } i, j \in [\![1, ..., N_1]\!] \\ \infty \text{ otherwise} \end{cases}$$

$$\text{and for insertion : } \mathbf{A}_{i,j} = \begin{cases} \|\mathbf{H}'_i\|_2 \text{ if } i = j, \text{ with } i, j \in [\![1, ..., N_2]\!] \\ \infty \text{ otherwise} \end{cases}$$

Final costs matrix

We then obtain a costs matrix $\tilde{\mathbf{C}}$ similar to those used in Bipartite matching.

$$\tilde{\mathbf{C}} = \begin{pmatrix} \mathbf{C} & \mathbf{D} \\ \mathbf{A} & 0 \end{pmatrix} \tag{3}$$

$$\tilde{\mathbf{C}} = \left(\begin{array}{ccc|ccc} \mathbf{c}_{1,1} & \cdots & \mathbf{c}_{1,N_2} & \mathbf{d}_1 & \cdots & \infty \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}_{N_1,1} & \cdots & \mathbf{c}_{N_1,N_2} & \infty & \cdots & \mathbf{d}_{N_1} \\ \hline \mathbf{a}_1 & \cdots & \infty & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \infty & \cdots & \mathbf{a}_{N_2} & 0 & \cdots & 0 \end{array}\right)$$

Output and Loss

we can then obtain a GED approximation value $\hat{\mathbf{s}}$ through discrete Optimal Transport by using **LSAP** on the $\tilde{\mathbf{C}}$ matrix :

$$\hat{\mathbf{s}}(G, G') = \mathbf{LSAP}(\tilde{\mathbf{C}}) \tag{4}$$

The Loss function used to train the architecture is a mean squared error :

$$\mathcal{MSE}_{Loss} = \frac{1}{|\mathcal{T}|} \sum_{(G,G') \in \mathcal{T}} \left\| \hat{\mathbf{s}}(G, G') - GED(G, G') \right\|_2^2 \tag{5}$$

With $\mathcal{T}$ the set of training pairs of graphs.

Metric properties conservation

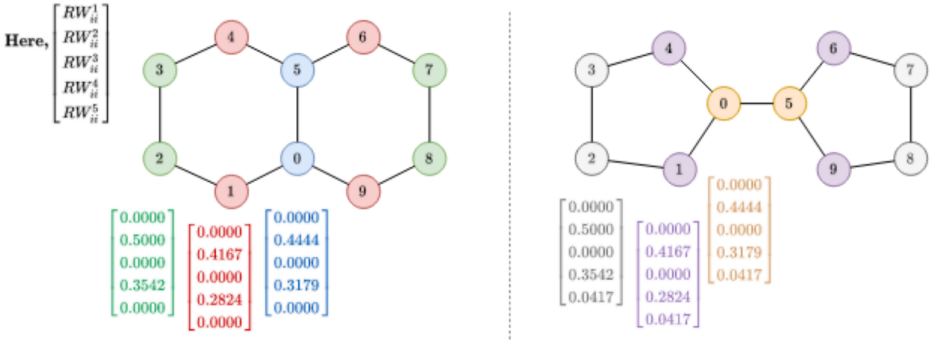
We want $\hat{\mathbf{s}}$ to respect the metric properties of GED. $\hat{\mathbf{s}} : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$ satisfying the following axioms for all graphs $x, y, z \in \mathcal{G}$:

1)The distance from a graph to itself is zero: $\hat{\mathbf{s}}(x, x) = 0$. Holds.
2)Positivity: If $x \neq y$, then $\hat{\mathbf{s}}(x, y) > 0$. Not respected.
3)Symmetry: $\hat{\mathbf{s}}(x, y) = \hat{\mathbf{s}}(y, x)$. Holds.
4)The triangle inequality: $\hat{\mathbf{s}}(x, z) \leq \hat{\mathbf{s}}(x, y) + \hat{\mathbf{s}}(y, z)$. Holds.

All those properties are respected during learning except for the Positivity due to the Graph isomorphism problem which is NP-Hard. It came from the expressivity bound of GIN (first-order Weisfeiler-Lehman test).
Therefore $\hat{\mathbf{s}}(G, G')$ is a pseudo-metric. Indeed it exists $G \neq G'$ for which $\hat{\mathbf{s}}(G, G') = 0$.

Positional encoding

Discriminative power of Random Walk for graphs not differentiated by 1-WL.



$$\mathbf{RW}_i^{(k)} = [(\mathbf{D}^{-1}\mathbf{A})_{(i,i)}, (\mathbf{D}^{-1}\mathbf{A})^2_{(i,i)}, ..., (\mathbf{D}^{-1}\mathbf{A})^{k-1}_{(i,i)}, (\mathbf{D}^{-1}\mathbf{A})^k_{(i,i)}] \quad (6)$$

Where $\mathbf{D}^{-1}\mathbf{A}$ is the random walk Laplacian.

Experiments and results

- AIDS: 490 000 graph pairs, max 10 nodes, 29 features.
- LINUX: 1 Million graph pairs, max 10 nodes, no features.
- IMDB: 2.25 Million graph pairs, max 100 nodes, no features.

Results are expressed with Root Mean Squared Error (RMSE) metric.

Methods	AIDS'	LINUX	IMDB
Branch	3.322	2.474	**6.875**
MIP-F2	2.929	1.245	82.124
GREED	0.796	0.415	49.81
H^2MN	0.994	0.734	86.077
GENN-A*	0.907	**0.267**	NA
GotSIM	0.996	0.574	37.831
SimGNN	1.037	0.666	66.250
Ours	0.799	0.474	31.354
Ours-RW	**0.698**	0.328	27.147

Conclusion?

- The supervised learning of the GED works
- Representations of the graphs and nodes are very important
- Two main blocks in the architecture, GNN for embedding node distributions, Euclidean distance and discrete Optimal transport for metric matching.
- LSAP brings explainability through the matching performed.
- Good Problem definition? Do not scale to larger graphs.
- Unsupervised learning? Other metric labels?

Bibliography I


Bai, Y., Ding, H., Bian, S., Chen, T., Sun, Y., and Wang, W. (2018).
Simgnn: A neural network approach to fast graph similarity computation.


Bougleux, S. and Brun, L. (2016).
Linear Sum Assignment with Edition.
Research report, Normandie Université ; GREYC CNRS UMR 6072.


Doan, K. D., Manchanda, S., Mahapatra, S., and Reddy, C. K. (2021).
Interpretable graph similarity computation via differentiable optimal alignment of node embeddings.
SIGIR.


Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. (2022).
Graph neural networks with learnable structural and positional representations.
In *International Conference on Learning Representations*.


Fischer, A., Suen, C. Y., Frinken, V., Riesen, K., and Bunke, H. (2015).
Approximation of graph edit distance based on hausdorff matching.
Pattern Recognition.


Lerouge, J., Abu-Aisheh, Z., Raveaux, R., Héroux, P., and Adam, S.
Exact Graph Edit Distance Computation Using a Binary Linear Program.
In *Structural, Syntactic, and Statistical Pattern Recognition 2016*.


Peyré, G. and Cuturi, M. (2018).
Computational optimal transport.

Bibliography II


Ranjan, R., Grover, S., Medya, S., Chakaravarthy, V., Sabharwal, Y., and Ranu, S. (2022).
GREED: A neural framework for learning graph distance functions.
NeurIPS.


Riesen, K. and Bunke, H. (2009).
Approximate graph edit distance computation by means of bipartite graph matching.
Image and Vision Computing, 27(7):950–959.


Wang, R., Zhang, T., Yu, T., Yan, J., and Yang, X. (2020).
Combinatorial learning of graph edit distance via dynamic embedding.


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


Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018).
How powerful are graph neural networks?


Zeina Abu-Aisheh, Romain Raveaux, J.-Y. R. P. M.
An exact graph edit distance algorithm for solving pattern recognition problems.
4th International Conference on Pattern Recognition Applications and Methods 2015.


Zhang, Z., Bu, J., Ester, M., Li, Z., Yao, C., Yu, Z., and Wang, C. (2021).
H2mn: Graph similarity learning with hierarchical hypergraph matching networks.
In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining*, KDD '21, page 2274–2284, New York, NY, USA. Association for Computing Machinery.

Thank you for your attention!