Graph Neural Networks and Optimal Transport for Distance Learning between Graphs

GT Graph

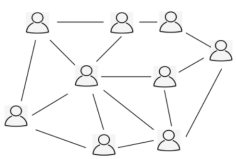
Aldo Moscatelli[1,2]
Tutors : Sébastien Adam[2],
Maxime Berar[2], Pierre Héroux[2].

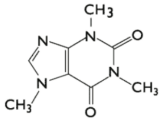
[1] Presentation author, aldo.moscatelli@univ-rouen.fr
[2] LITIS Lab, University of Rouen Normandy

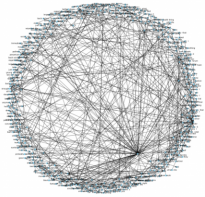
May 23, 2024

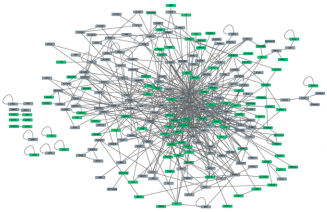
Représentation de données par des graphes

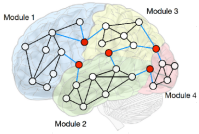
Un "langage" très général pour décrire et analyser des entités en interactions.



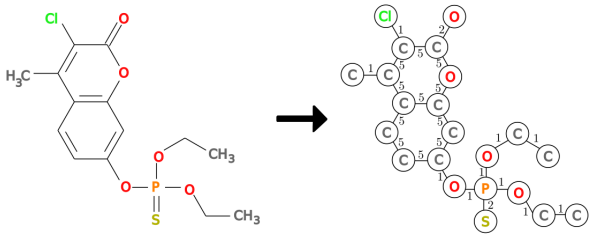




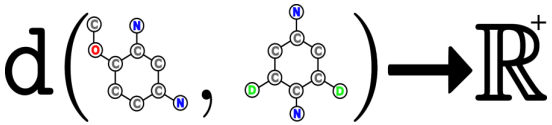




Graphes et distance



Challenge : les graphes sont des objets non Euclidiens et le produit scalaire n'est pas défini pour les graphes
Comment définir une distance entre des graphes ?

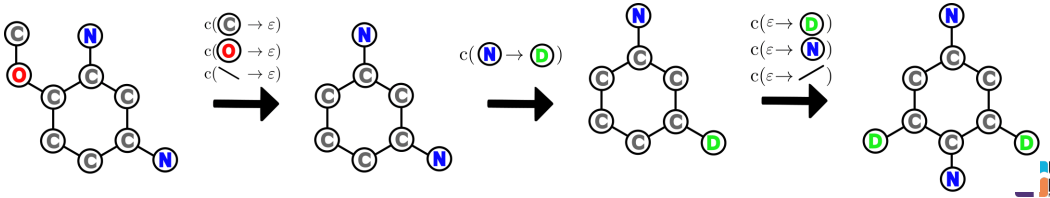


Graph Edit Distance (GED)

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

avec (e_i) les opérations du chemin d'édition de G_1 à G_2.

Ensemble des opérations d'édition
- **node insertion**
- **node deletion**
- **node substitution**
- **edge insertion**
- **edge deletion**
- **edge substitution**



Approches traditionnelles

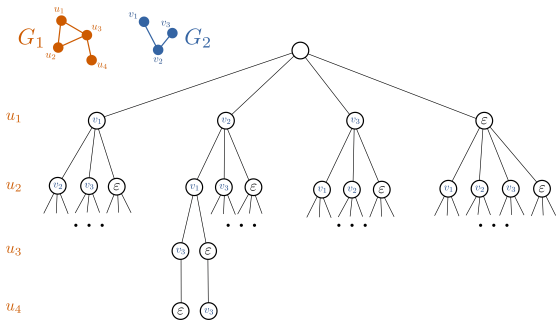
Méthodes exactes :

- Algorithme $\mathbf{A}^*$ avec heuristique
- Algorithme depth-first search
- Résolution du QAP avec un ILP solver (F1, F2)

problème NP-Hard

Méthodes d'approximation :
Méthodes basées sur les nœuds

- Relaxation du QAP en LSAP
- Bipartite matching (BPGED)
- Hausdorff matching



Peut-on apprendre la GED ?

Pourquoi ?

- Pour contourner la complexité des méthodes pré-existantes
- Pour avoir de meilleures approximations

Comment ?

- Architecture siamoise pour traiter les paires de graphes
- Avec des Graph Neural Networks (GNNs) pour la représentation des nœuds
- Transport Optimal/LSAP pour faire le matching des distributions des représentations des nœuds et inférer le coût d'édition minimal / la distance

Méthodes pré-existantes utilisant du Deep Learning (SOTA) :

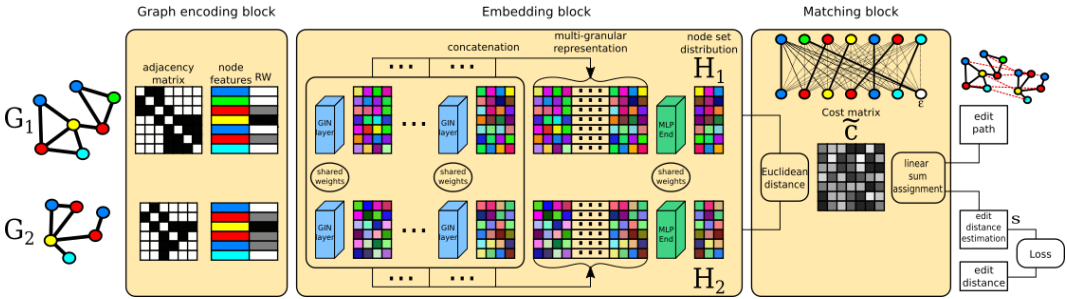
- SimGNN/GotSim/Greed/H2MN/GENN-A*...

UNIVERSITÉ DE ROUEN NORMANDIE

Comparaison SOTA

	SimGNN	GotSim	Greed	H2MN	GENN-A*	**GNOME(ours)**
Explicabilité	✗	✔	✗	✗	✔	✔
Node-level	✔	✔	✗	✗	✔	✔
Metric properties	✗	✗	✔	✗	✗	✔
Scalability	✔	✔	✔	✔	✗	✔

Présentation de GNOME(Graph NOde Matching for Edit distance)



Encoding Block

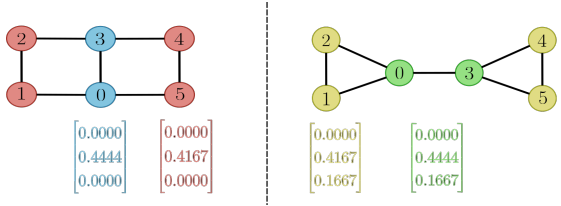


Figure: Illustration du descripteur Random Walk de taille 3. Ce descripteur permet de distinguer les deux graphes alors que le test de Weisfeiler-Lehmann d'ordre 1 ne peut pas les séparer.

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

Où $\mathbf{D}^{-1}\mathbf{A}$ est le Laplacien Random-Walk.

Embedding block

Graph Isomorphism Network (GIN) est utilisé pour apprendre une représentation des nœuds des graphes en aggrégeant chaque voisinage de chaque nœud à chaque couche :

$$\mathbf{h}_v^{(l+1)} = MLP^{(l)} \left(\left(1 + \epsilon^{(l)}\right) .\mathbf{h}_v^{(l)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(l)} \right) \tag{2}$$

où l est la l-ième couche de GIN, $\mathcal{N}(v)$ le voisinage d'ordre 1 du nœud v, $\epsilon^{(l)}$ un paramètre apprenable et MLP un multi-layer perceptron avec deux couches.
Nous obtenons finalement après L couches de GIN, la représentation finale des nœuds :

$$\mathbf{H}_v = MLP^{(end)} \left(\mathbf{h}_v^{(1)} | \mathbf{h}_v^{(2)} | ... | \mathbf{h}_v^{(L-1)} | \mathbf{h}_v^{(L)} \right) \tag{3}$$

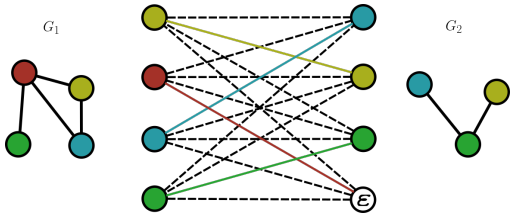
Matching Block

Pour opérer le matching des deux distributions, nous utilisons un Linear Sum Assignment Problem (LSAP) solver.
En introduisant une matrice de permutation $\mathbf{X} = (x_{i,j})$, le LSAP se défini par

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

$$\mathbf{C} = \begin{pmatrix} \mathbf{S} & \mathbf{D} \\ \mathbf{I} & 0 \end{pmatrix} \quad (4)$$

$$\tilde{\mathbf{C}} = \begin{pmatrix} \mathbf{S} & \tilde{\mathbf{D}} \end{pmatrix} \quad (5)$$



Coûts de substitution

Soit une paire de graphes (G, G') de taille (N_1, N_2).Nous obtenons comme défini par l'eq. (2), deux ensembles de représentation de nœuds $\mathbf{H}, \mathbf{H}'$.
Nous pouvons maintenant calculer une matrice de coûts en utilisant la distance Euclidienne entre les représentations.

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

Les coûts de $\mathbf{S}$ correspondent aux coûts de substitution de la GED entre les nœuds de G et ceux de G'.

Les coûts de suppression (ou insertion)

On peut aussi utiliser la distance Euclidienne pour définir les coûts de suppression. Dans ce cas, la norme du vecteur de représentation est utilisée.

$$\text{Suppression : } \tilde{\mathbf{D}} = \begin{pmatrix} \tilde{d}_{1,1} & \cdots & \tilde{d}_{1,N_1-N_2} \\ \vdots & \ddots & \vdots \\ \tilde{d}_{N_1,1} & \cdots & \tilde{d}_{N_1,N_1-N_2} \end{pmatrix}, \tilde{d}_{i,j} = \|\mathbf{H}_{1,i}\|_2 \,, \tag{6}$$

Output et Loss

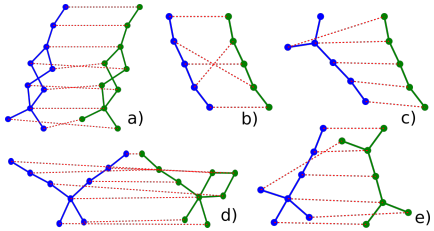
On peut ensuite récupérer le matching et inférer une valeur approximée **s** de la GED en utilisant le **LSAP** sur la matrice $\tilde{\mathbf{C}}$:

$$\mathbf{s}(G, G') = \mathbf{LSAP}(\tilde{\mathbf{C}}) \tag{7}$$

La fonction de Loss utilisée pour l'entrainement est la MSE(Mean Squared Error) :

$$\mathcal{MSE}_{Loss} = \frac{1}{|\mathcal{T}|} \sum_{(G,G') \in \mathcal{T}} \left\| \mathbf{s}(G, G') - GED(G, G') \right\|_2^2 \tag{8}$$

Avec
$\mathcal{T}$ l'ensemble des paires de graphes d'entraînement.



Conservation des propriétés métriques

Nous voulons que **s** respecte les propriétés de distance de la GED $\mathbf{s} : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$ satisfaisant les axiomes suivants, pour tous les graphes $x, y, z \in \mathcal{G}$:

1) La distance d'un graphe à lui même est 0 : $\mathbf{s}(x, x) = 0$. Respectée.
2) Positivité : Si $x \neq y$, alors $\mathbf{s}(x, y) > 0$. Non respectée.
3) Symétrie : $\mathbf{s}(x, y) = \mathbf{s}(y, x)$. Respectée.
4) L'inégalité triangulaire: $\mathbf{s}(x, z) \leq \mathbf{s}(x, y) + \mathbf{s}(y, z)$. Respectée.

Toutes ces propriétés sont respectées pendant l'apprentissage sauf la Positivité à cause du problème d'isomorphisme de graphes qui est NP-Hard. Celà vient de limitations des modèles de deep learning sur graphes (ici GIN, 1-WL test equivalent).
De ce fait $\mathbf{s}(G, G')$ est une pseudo-metrique. En effet il existe $G \neq G'$ pour lesquels $\mathbf{s}(G, G') = 0$.

UNIVERSITÉ DE ROUEN NORMANDIE

Expériences et résultats

- AIDS : 490 000 paires de graphes , max 10 nœuds, 29 features.
- LINUX : 1 Million de paires de graphes, max 10 nœuds, pas de features.
- IMDB : 2.25 Million de paires de graphes, max 100 nœuds, pas de features.

Settings :

- 200 epoch
- batch 32
- learning rate 10^{-4}
- 8 GIN layers + skip connections
- training Loss: MSE
- 12 steps RW

UNIVERSITÉ DE ROUEN NORMANDIE

Dataset	Metrics	GED			Ranking				Time
		MAE	MSE	MSE(10^{-3})	ρ	τ	p@10	p@20	(s/100 p)
Linux	SimGNN	0.489	0.443	2.172	0.939	0.830	94.2%	93.3%	0.244
	GREED	0.318	0.172	0.914	-	-	-	-	0.007
	GotSIM	-	0.329	4.25	0.92	0.89	86%	-	-
	H^2MN	0.534	0.538	1.630	0.984	-	95.3%	-	0.087
	GEDGNN	**0.094**	-	-	0.963	0.903	96.2%	97.6%	0.380
	TaGSim	0.391	-	5.278	0.941	0.834	81.6%	86.7%	0.117
	Noah	1.747	-	-	0.874	0.802	90.9%	93.6%	77.237
	GENN-A^*	-	**0.071**	**0.324**	**0.991**	-	96.2%	-	217.7
	GNOME	0.214±0.011	0.104±0.015	0.572±0.075	0.987	0.932	**98.3%**	99.1%	0.2653
	GNOME-RW	0.195±0.006	0.077±0.004	0.423±0.089	0.989	**0.936**	98.1%	**99.3%**	0.2653
AIDS	SimGNN	0.816	1.075	2.238	0.843	0.690	42.1%	51.4%	0.283
	GREED	0.629	0.634	1.334	-	-	-	-	0.005
	GotSIM	-	0.992	2.36	0.86	0.72	87%	-	-
	H^2MN	0.777	0.988	1.913	0.877	-	51.7%	-	0.095
	GEDGNN	0.773	-	-	0.876	0.751	71.6%	77.9%	0.408
	TaGSim	0.841	-	9.827	0.688	0.527	64.6%	32.2%	0.123
	Noah	1.542	4.675	-	0.734	0.560	80.9%	81.2%	168.390
	GENN-A^*	-	0.823	**0.635**	0.959	-	**87.1%**	-	1332.3
	GNOME	0.555±0.009	0.490±0.017	0.990±0.023	0.949	0.846	80.4%	84.5%	0.265
	GNOME-RW	**0.508±0.008**	**0.407±0.014**	0.849±0.032	**0.959**	**0.863**	81.3%	**86.7%**	0.265
IMDB	SimGNN	28.082	4389.06	5.618	0.878	0.770	75.9%	77.7%	0.258
	GREED	3.612	45.347	1.243	-	-	-	-	0.006
	GotSIM	-	4424.9	5.92	0.85	0.80	73%	-	-
	H^2MN	28.486	7409.25	0.744	0.912	-	86.1%	-	0.083
	GEDGNN	-	-	-	-	-	-	-	-
	TaGSim	-	-	35.69	0.958	0.926	-	**98.6%**	0.113
	Noah	3.755	55.636	-	0.810	0.716	35.4%	40.5%	5201.6
	GENN-A^*	-	-	-	-	-	-	-	-
	GNOME	2.976±0.004	**33.433±0.054**	0.831±0.021	0.994	0.953	87.7%	87.8%	0.272
	GNOME-RW	**2.920±0.007**	33.769±0.072	**0.722±0.053**	**0.995**	**0.955**	**88.2%**	88.6%	0.272

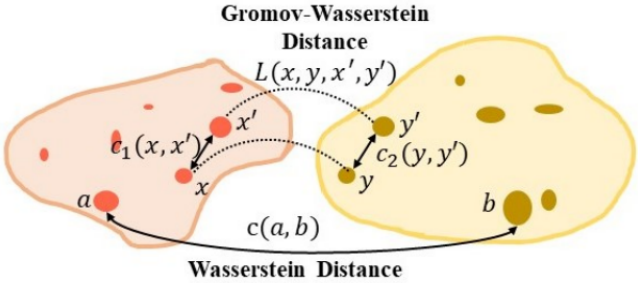
Limitations et Perspectives

- GIN est 1-WL $\rightarrow$ nodes symétries
- RW en potitionnal encoding, inclues dans le language 3-WL
- Pas de prise en compte des arcs directement dans le matching
- Pas de prise en compte directe des coûts des opérations d'éditions
- Les valeurs de regressions ne sont pas allignées avec les coût des paths
- Le LSAP est une linéarisation du QAP, une autre forme de simplification?

Optimal transport

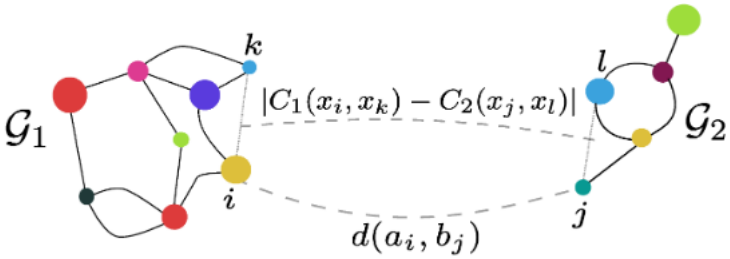
Prendre en compte les arcs dans le matching → QAP → Transport Optimal, Distance de Gromov-Wasserstein + partial + fused. On passe de l'espace des matrices de permutations aux matrices bi-stochastiques.

$$QAP\_GED(G_1, G_2) = \min_{\mathbf{x}} \left\{ \frac{1}{2}\mathbf{x}^T \Delta \mathbf{x} + \mathbf{c}^T \mathbf{x} \right\} \tag{9}$$

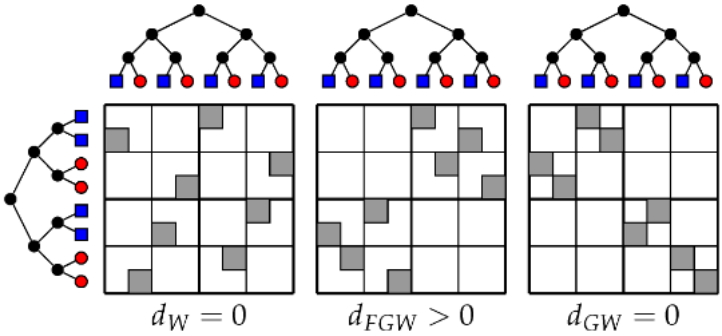


OT Fused Gromov-Wasserstein

$$\mathbf{FGW} \in \min_{\mathbf{T}} (1-\alpha)\langle \mathbf{T},\ \mathbf{M}\rangle_F + \alpha \sum_{i,j,k,l} \mathcal{L}(\mathbf{C_1}_{i,k}, \mathbf{C_2}_{j,l})\mathbf{T}_{i,j}\mathbf{T}_{k,l}$$
$$\text{s.t. } \mathbf{T1} = \mathbf{p}$$
$$\mathbf{T}^T\mathbf{1} = \mathbf{q}$$
$$\mathbf{T} \geq 0$$



Différentes distances = Différents matchings



Mais pour ce genre de matching, nous avons besoin de représentations du graphe node level et edge level, ça tombe bien on en a une sous le coude :) (cf présentation G2N2 de Jason)

"Fused Gromov-Wasserstein Distance for Structured Objects" Titouan Vayer,Laetitia Chapel et al.

Regularisations du chemin d'édition

Pour prendre en compte les coûts des opérations d'editions et réaligner la valeur de régression du modèle avec le coût fourni par le chemin d'édition equivalent on peut régulariser la Loss du modèle.

$$QAP\_GED(G_1, G_2) = \min_{\mathbf{x}} \left\{ \frac{1}{2}\mathbf{x}^T \Delta \mathbf{x} + \mathbf{c}^T \mathbf{x} \right\} \tag{10}$$

En précalculant Δ et $\mathbf{c}$ qui dépendent des coûts d'éditions, de la structure des graphes et de son signal, on peut utiliser cette information pour régulariser la Loss en évaluant les chemin d'editions fourni par le modèle pendant l'apprentissage.

Pour aller plus loin

- Distance task-driven, regression/classif, contrastive learning?
- Le matching est très couteux (très lent)
- Version parallèle + différentiable d'EMD
- Peut on apprendre le matching?
- Redéfinir le concept de similarité entre graphes en sortant du carcan de la GED

Merci pour votre attention !
Avez-vous des questions ?

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.

Burkard, R. E. and Derigs, U. (1980).
The Linear Sum Assignment Problem, pages 1–15.
Springer Berlin Heidelberg, Berlin, Heidelberg.

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

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.

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

UNIVERSITÉ DE ROUEN NORMANDIE

Bibliography II

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.

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 and Data Mining*, KDD '21, page 2274–2284, New York, NY, USA. Association for Computing Machinery.