Grammatical Graph Neural Networks and more

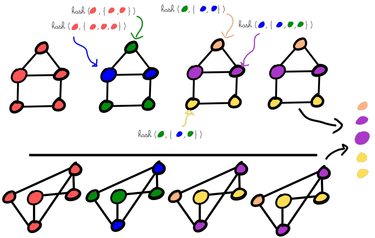
Jason Piquenot

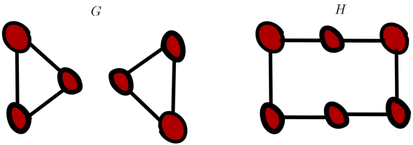
Supervisors : Sébastien Adam Romain Raveaux
Co-supervisors : Maxime Bérar Pierre Héroux Jean-Yves Ramel

(ANR-21-CE-0025)

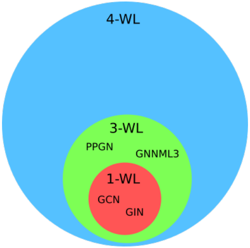
8^{th} december 2023

Weisfeiler-lehman hierarchy





Algorithm k-*WL* ($k \geq 2$)
Input: $G = (V, E, X_V)$
1. $c^0_{\vec{v}} \leftarrow \text{hash}(G[\vec{v}])$ for all $\vec{v} \in V^k$
2. **repeat**
3. $\quad c^\ell_{\vec{v},i} \leftarrow \{\!\{c^{\ell-1}_w : w \in \mathcal{N}_i(\vec{v})\}\!\}\ \forall v \in V^k, i \in [k]$
4. $\quad c^\ell_{\vec{v}} \leftarrow \text{hash}(c^{\ell-1}_{\vec{v}}, c^\ell_{\vec{v},1}, \dots c^\ell_{\vec{v},k})\ \forall \vec{v} \in V^k$
5. **until** $(c^\ell_{\vec{v}})_{\vec{v} \in V^k} == (c^{\ell-1}_{\vec{v}})_{\vec{v} \in V^k}$
6. **return** $\{\!\{c^\ell_{\vec{v}} : \vec{v} \in V^k\}\!\}$



Contributions

New GNN design framework based on Context Free Grammar (CFG).

Grammatical Graph Neural Network (G^2N^2) a 3-WL GNN.

3-WL spectral response.

Matlang

In [Brijder et al., 2019] they introduced MATLANG a matrix language.

Matlang

$\mathrm{ML}(\mathcal{L})$ is a matrix language with an allowed operation set $\mathcal{L} = \{op_1, \dots, op_n\}$, where $op_i \in \{\cdot, +, {}^{\mathbf{T}}, \mathrm{diag}, \mathrm{Tr}, 1, \odot, \times, f\}$.

Sentence

$e(X) \in \mathbb{R}$ is a sentence in $\mathrm{ML}(\mathcal{L})$ if it consists of any possible consecutive operations in $\mathcal{L}$, operating on a given matrix X and resulting in a scalar value.

As an example, $1^{\mathbf{T}}(X \odot \mathrm{diag}(1))1$ is a sentence in $\mathrm{ML}\left({}^{\mathbf{T}}, 1, \odot, \cdot, \mathrm{diag}\right)$ that computes trace of square matrix X.

Graphs equivalence in MATLANG

ML $(\mathcal{L})$-*equivalent* for matrices

Two matrices A and B in $\mathcal{M}_{m,n}(\mathbb{R})$ are said to be ML $(\mathcal{L})$-*equivalent*, denoted by $A \equiv_{\mathrm{ML}(\mathcal{L})} B$, if and only if $e(A) = e(B)$ for all sentences in ML $(\mathcal{L})$.

ML $(\mathcal{L})$-*equivalent* for graphs

Two graphs $\mathcal{G}$ and $\mathcal{H}$ of the same order are said to be ML $(\mathcal{L})$-*equivalent*, denoted by $\mathcal{G} \equiv_{\mathrm{ML}(\mathcal{L})} \mathcal{H}$, if and only if their adjacency matrices are ML $(\mathcal{L})$-*equivalent*.

Link between WL and MATLANG

In [Geerts and Reutter, 2021], they proved the following theorems for the sets of operations $\mathcal{L}_1 = \{\cdot, {}^{\mathbf{T}}, 1, \mathrm{diag}\}$ and $\mathcal{L}_3 = \{\cdot, {}^{\mathbf{T}}, 1, \mathrm{diag}, \odot\}$.

1-WL equivalence

Two adjacency matrices are indistinguishable by the 1-WL test if and only if $e(A_G) = e(A_H)$ for all $e \in \mathcal{L}_1$. Hence, all possible sentences in $\mathcal{L}_1$ are the same for 1–WL-*equivalent* adjacency matrices. Thus,

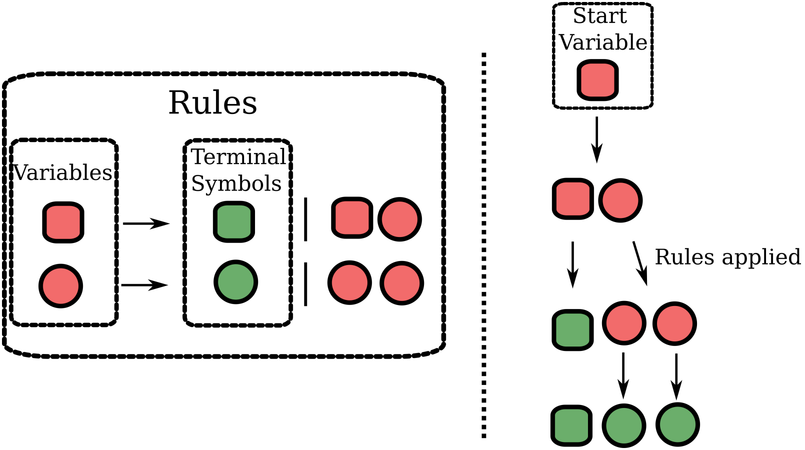
$$A_{\mathcal{G}} \equiv_{\text{1-WL}} A_{\mathcal{H}} \iff A_{\mathcal{G}} \equiv_{\mathrm{ML}(\mathcal{L}_1)} A_{\mathcal{H}}$$

3-WL equivalence

Two adjacency matrices are indistinguishable by the 3-WL test if and only if $e(A_G) = e(A_H)$ for all $e \in \mathcal{L}_3$. Hence, all possible sentences in $\mathcal{L}_3$ are the same for 3–WL-*equivalent* adjacency matrices. Thus,

$$A_{\mathcal{G}} \equiv_{\text{3-WL}} A_{\mathcal{H}} \iff A_{\mathcal{G}} \equiv_{\mathrm{ML}(\mathcal{L}_3)} A_{\mathcal{H}}$$

Context Free Grammar



A framework to translate a language into GNN

We proposed the following framework to translate a language into a GNN in 3 steps:

Define the exhaustive CFG that generates the language

Reduce the exhaustive CFG

Translate the variables and rules of the reduced CFG into GNN input and model layer

Our framework applied on ML $(\mathcal{L}_1)$

Language fragment

$\mathrm{ML}(\mathcal{L}_1)$
$(\mathcal{L}_3) = \{\cdot, {}^{\mathrm{T}}, \mathbb{1}, \mathrm{diag}\ \}$

→

Exhaustive CFG

$G_{\mathcal{L}_1}$

$S \to (V_r)(V_c) | \mathrm{diag}(S) | SS$
$V_c \to MV_c | (V_r)^{\mathrm{T}} | V_c S | \mathbb{1}$
$V_r \to V_r M | (V_c)^{\mathrm{T}} | SV_r$
$M \to MM | (M)^{\mathrm{T}} | \mathrm{diag}(V_c) | (V_c)(V_r) | A$

→

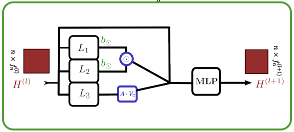
Reduced CFG

r-$G_{\mathcal{L}_1}$

$V_c \to\ V_c \odot V_c \mid AV \mid \mathbb{1}$

→

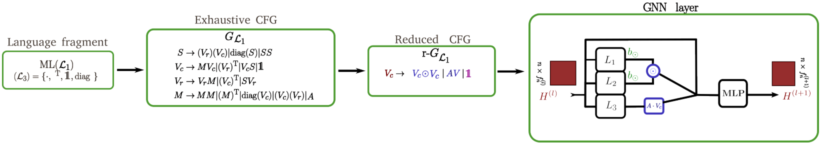
GNN layer

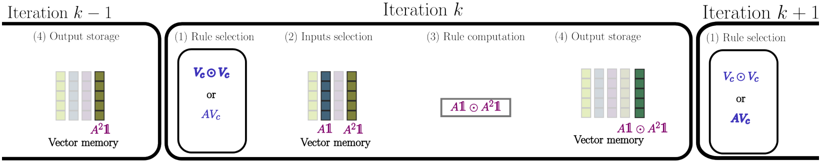


Example of the generation of the word $(1)^{\mathrm{T}}A1$ with ML $(\mathcal{L}_1)$.

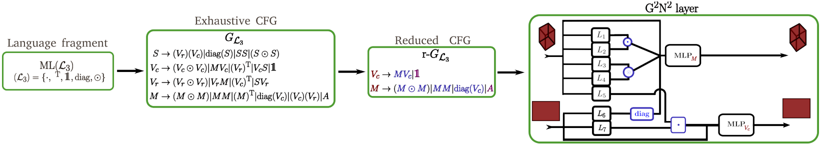
$$S \xrightarrow[S \to V_r V_c]{} V_r V_c \xrightarrow[V_r \to (V_c)^{\mathrm{T}}]{} (V_c)^{\mathrm{T}} V_c \xrightarrow[V_c \to MV_c]{} (V_c)^{\mathrm{T}} M V_c \xrightarrow[V_c \to \mathbb{1}]{} (V_c)^{\mathrm{T}} M \mathbb{1} \xrightarrow[V_c \to \mathbb{1}]{} (\mathbb{1})^{\mathrm{T}} M \mathbb{1} \xrightarrow[M \to A]{} (\mathbb{1})^{\mathrm{T}} A \mathbb{1}$$

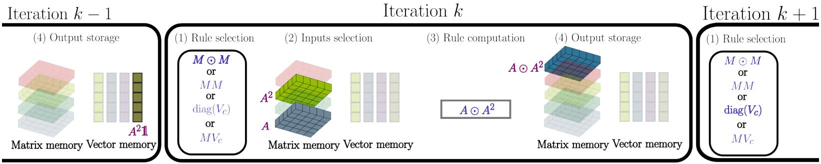
Our framework applied on ML ($\mathcal{L}_1$)



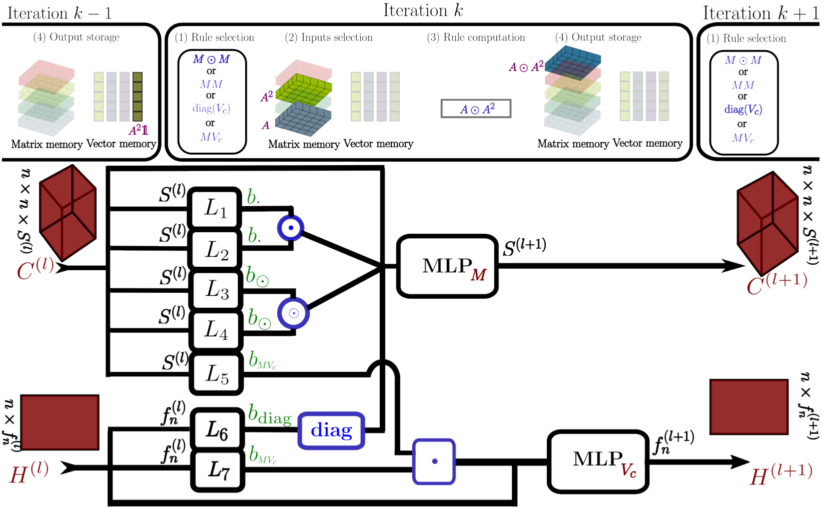


Our framework applied on ML ($\mathcal{L}_3$)



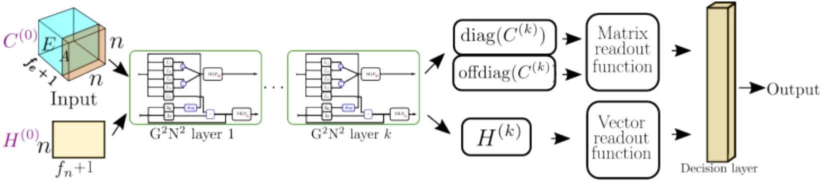


parallelised rule generation and G^2N^2



G^{2}N^2 a 3-WL GNN

G^{2}N^2 architecture :



Theorem (G^{2}N^2 in the WL hierarchy)

G^2N^2 is as expressive as 3-WL.

3-WL spectral response

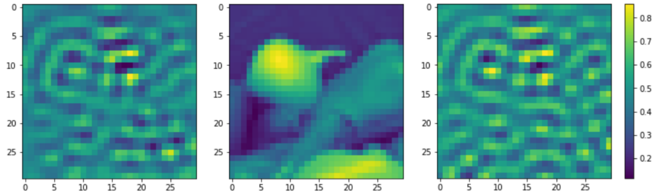
Theorem (3-WL spectral response)

3-WL can approximate low-pass, high-pass and band-pass filter in the spectral domain.

Table: R^2 score on spectral filtering node regression problems. Results are median of 10 different runs.

Method	Low-pass	High-pass	Band-pass
CHEBNET	0.9995	0.9901	0.8217
GNNML3	0.9995	0.9909	0.8189
PPGN	0.9991	**0.9925**	0.1041
G^2N^2	**0.9996**	**0.9994**	0.8206

On the left G^2N^2's prediction, on the center the input and on the right, the target



Regression tasks

Table: Results on QM9 dataset focusing on the best methods. The metric is MAE, the lower, the better.

Target	PPGN	G^2N^2
μ	0.0934	**0.0703**
α	0.318	**0.127**
ϵ_{homo}	0.00174	**0.00172**
ϵ_{lumo}	0.0021	**0.00153**
$\Delta\epsilon$	0.0029	**0.00253**
R^2	3.78	**0.342**
ZPVE	0.000399	**0.0000951**
U_0	0.022	**0.0169**
U	0.0504	**0.0162**
H	0.0294	**0.0176**
G	0.024	**0.0214**
C_v	0.144	**0.0429**
T / ep	129 s	98 s

PPGN	G^2N^2
0.231	**0.102**
0.382	**0.196**
0.00276	**0.0021**
0.00287	**0.00211**
0.0029	**0.00287**
16.07	**1.19**
0.00064	**0.0000151**
0.234	**0.0502**
0.234	**0.0503**
0.229	**0.0503**
0.238	**0.0504**
0.184	**0.0707**
131 s	57 s

Classification tasks

Table: Results of G^2N^2 on TUD dataset compared to the best competitor. The metric is accuracy, the higher, the better.

Dataset	G^2N^2	rank	Best GNN competitor
MUTAG	92.5±5.5	1(1)	92.2±7.5
PTC	72.3±6.3	1(1)	68.2±7.2
Proteins	80.1±3.7	1(1)	77.4±4.9
NCI1	82.8±0.9	5(3)	83.5±2.0
IMDB-B	76.8±2.8	3(2)	77.8±3.3
IMDB-M	54.0±2.9	2(2)	54.3±3.3

Conclusion

General framework to design GNN from language fragment

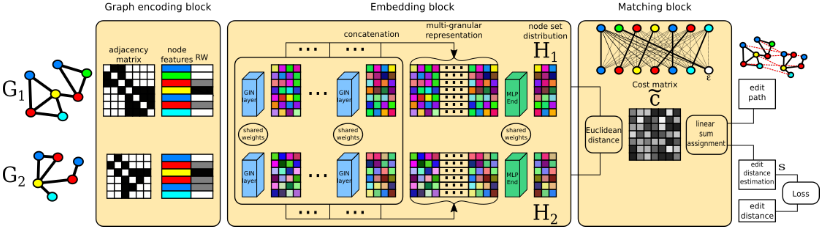
3-**WL** GNN resulting from our framework

3-WL spectral response

Perspective

applying our framework on language with other desired expressive power.

Graph NOde Matching for Edit distance(GNOME)



Embedding block

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

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

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 :

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

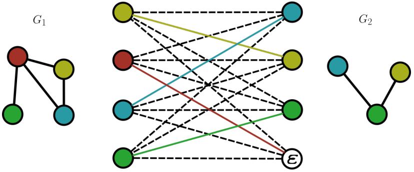
Matching Block

To proceed nodes distribution's matching an LSAP solver is used (injective case of Wasserstein distance)
By introducing permutation matrix $\mathbf{X} = (x_{i,j})$ we can define LSAP as follow :

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

$$\mathbf{C} = \begin{pmatrix} \mathbf{S} & \mathbf{D} \\ \mathbf{I} & 0 \end{pmatrix} \tag{3}$$

$$\tilde{\mathbf{C}} = \begin{pmatrix} \mathbf{S} & \tilde{\mathbf{D}} \end{pmatrix} \tag{4}$$



Substitution costs

Let be a graph pair (G, G') of size (N_1, N_2). We obtain as defined in (3), 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'.

Deletion (or insertion) costs

We can also use the Euclidean distance to define deletion costs. In this case, the norm of the embedding vector is used to represent its cost of deletion.

$$\text{Deletion: } \tilde{\mathbf{D}} = \begin{pmatrix} \tilde{d}_{1,1} & \cdots & \tilde{d}_{1,N_1-N_2} \\ \vdots & \ddots & \vdots \\ \tilde{d}_{N_1,1} & \cdots & \tilde{d}_{N_1,N_1-N_2} \end{pmatrix}, \tilde{d}_{i,j} = \left\| \mathbf{H}_i \right\|_2 , \tag{5}$$

Metric properties conservation

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

1)The distance from a graph to itself is zero: $\mathbf{S}(x,x) = 0$. Holds.
2)Positivity: If $x \neq y$, then $\mathbf{S}(x,y) > 0$. Not respected.
3)Symmetry: $\mathbf{S}(x,y) = \mathbf{S}(y,x)$. Holds.
4)The triangle inequality: $\mathbf{S}(x,z) \leq \mathbf{S}(x,y) + \mathbf{S}(y,z)$. 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 expressiveness bound of GIN (first-order Weisfeiler-Lehman test).
Therefore $\mathbf{S}(G,G')$ is a pseudo-metric. Indeed it exists $G \neq G'$ for which $\mathbf{S}(G,G') = 0$.

Triangular inequality for LSAP

Let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be three graphs. Without loss of generality let's assume $|\mathcal{X}| > |\mathcal{Y}| > |\mathcal{Z}|$. There exist two permutation matrices $P^*_{\mathcal{X},\mathcal{Y}}$ and $P^*_{\mathcal{Y},\mathcal{Z}}$ solutions of $\mathrm{LSAP}\left(\tilde{C}_{\mathcal{X},\mathcal{Y}}\right)$ and $\mathrm{LSAP}\left(\tilde{C}_{\mathcal{Y},\mathcal{Z}}\right)$.
Both matrices can be decomposed as:

$$P^*_{\mathcal{X},\mathcal{Y}} = \begin{pmatrix} S_{\mathcal{X},\mathcal{Y}} & \tilde{D}_{\mathcal{X},\mathcal{Y}} \end{pmatrix}, \quad P^*_{\mathcal{Y},\mathcal{Z}} = \begin{pmatrix} S_{\mathcal{Y},\mathcal{Z}} & \tilde{D}_{\mathcal{Y},\mathcal{Z}} \end{pmatrix}.$$

Let's construct $\tilde{P}$, the following permutation matrix

$$\tilde{P} = \begin{pmatrix} S_{\mathcal{X},\mathcal{Y}} S_{\mathcal{Y},\mathcal{Z}} & \tilde{D}_{\mathcal{X},\mathcal{Y}} \| S_{\mathcal{X},\mathcal{Y}} \tilde{D}_{\mathcal{Y},\mathcal{Z}} \end{pmatrix}$$

We have by definition and construction that

$$\mathrm{LSAP}\left(\tilde{C}_{\mathcal{X},\mathcal{Z}}\right) \leqslant \sum_{i=1}^{|\mathcal{X}|} \sum_{j=1}^{|\mathcal{Z}|} \left(\tilde{C}_{\mathcal{X},\mathcal{Z}}\right)_{i,j} \left(\tilde{P}\right)_{i,j} \leqslant \mathrm{LSAP}\left(\tilde{C}_{\mathcal{X},\mathcal{Y}}\right) + \mathrm{LSAP}\left(\tilde{C}_{\mathcal{Y},\mathcal{Z}}\right).$$

Brijder, R., Geerts, F., den Bussche, J. V., and Weerwag, T. (2019).
On the expressive power of query languages for matrices.
ACM Trans. Database Syst., 44(4):15:1–15:31.

Geerts, F. and Reutter, J. L. (2021).
Expressiveness and approximation properties of graph neural networks.
In *International Conference on Learning Representations*.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019).
How powerful are graph neural networks?
In *International Conference on Learning Representations*.

Thank you for your attention.

Our paper is available here.

Grammatical Graph Neural Network

https://arxiv.org/abs/2303.01590