Gaussian process regression for high dimensional graph inputs

Raphaël Carpintero Perez

Sébastien Da Veiga
Josselin Garnier
Brian Staber

12/10/2023

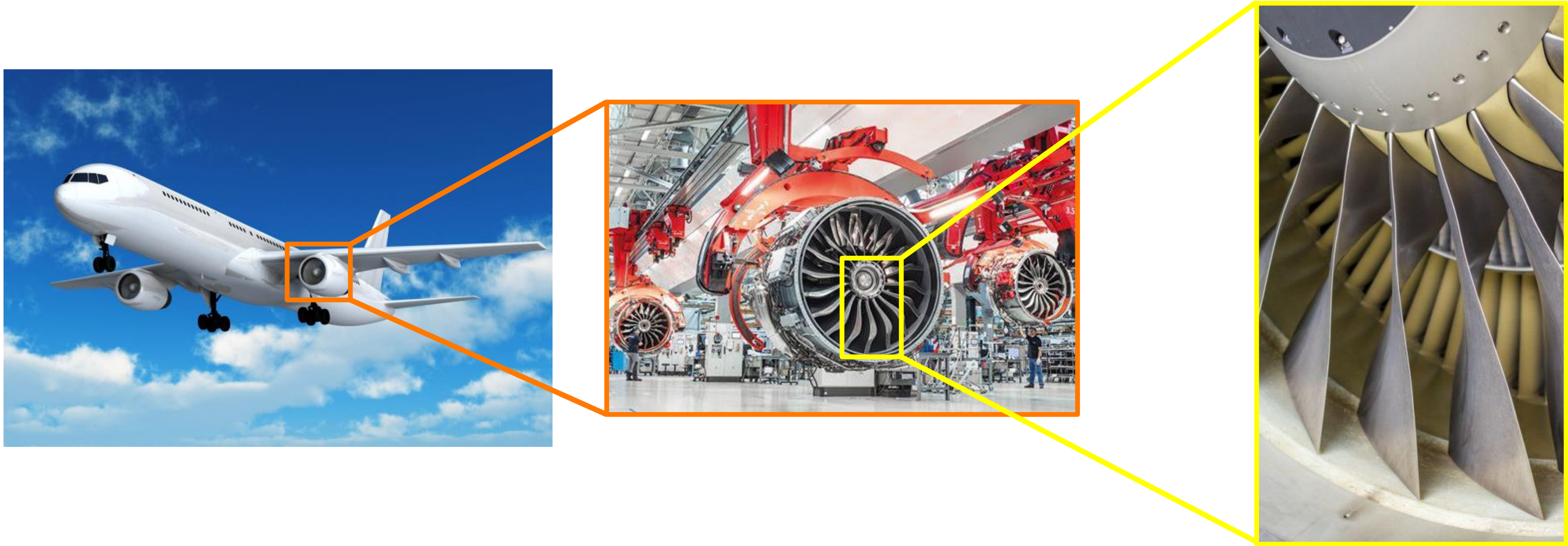
SAFRAN

CMAP

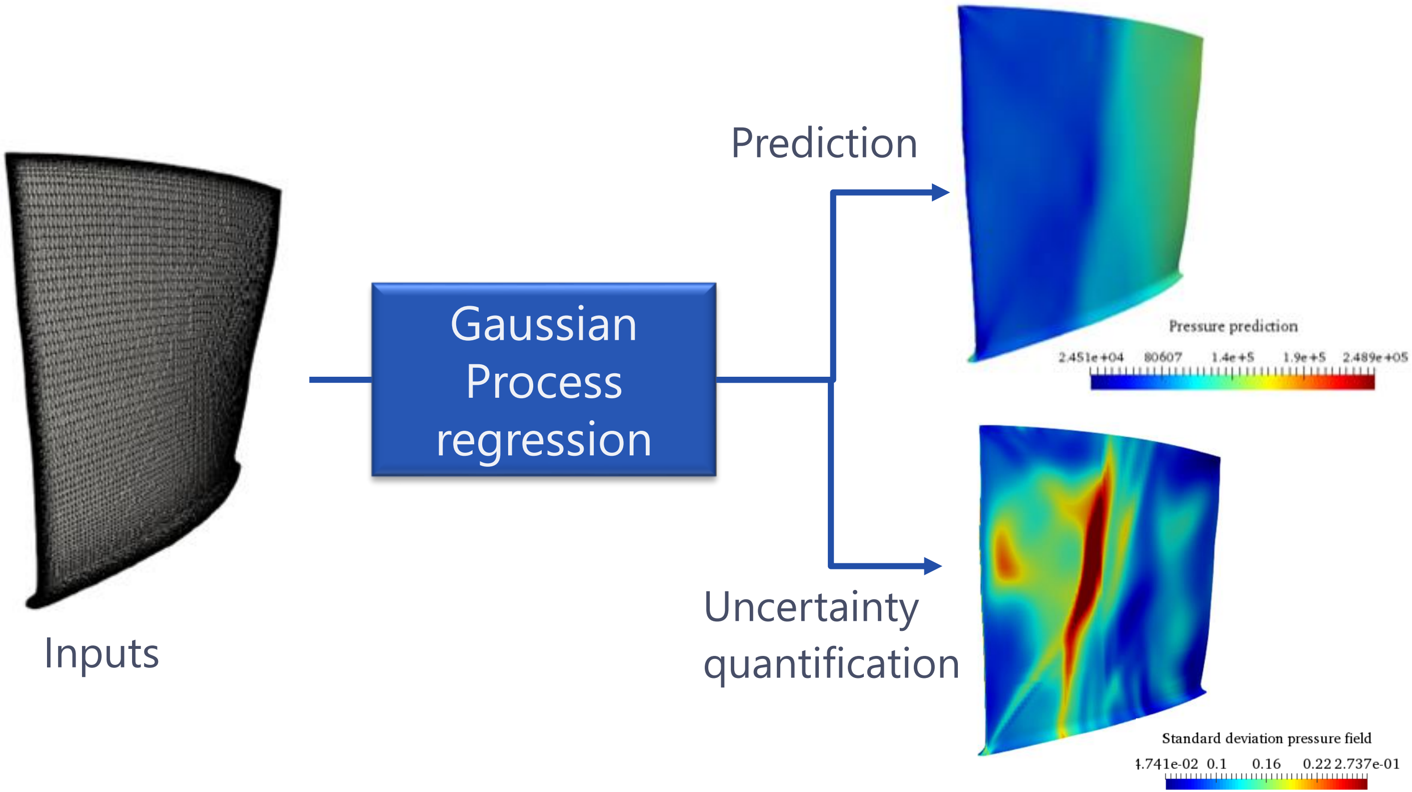
SAFRAN

Introduction

Objectives



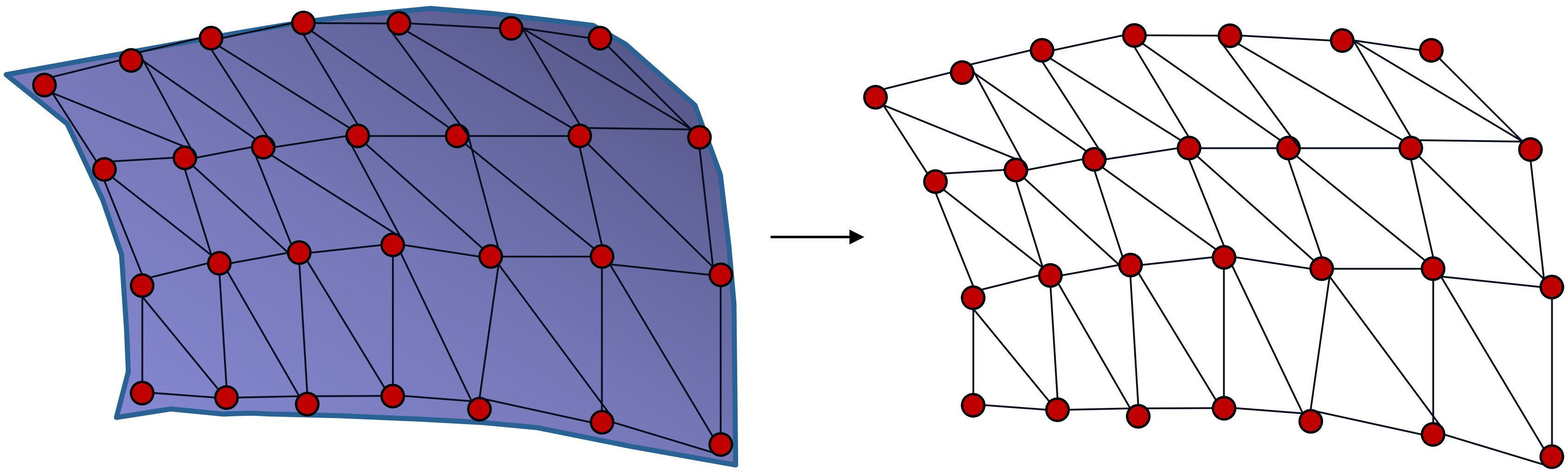
Objectives



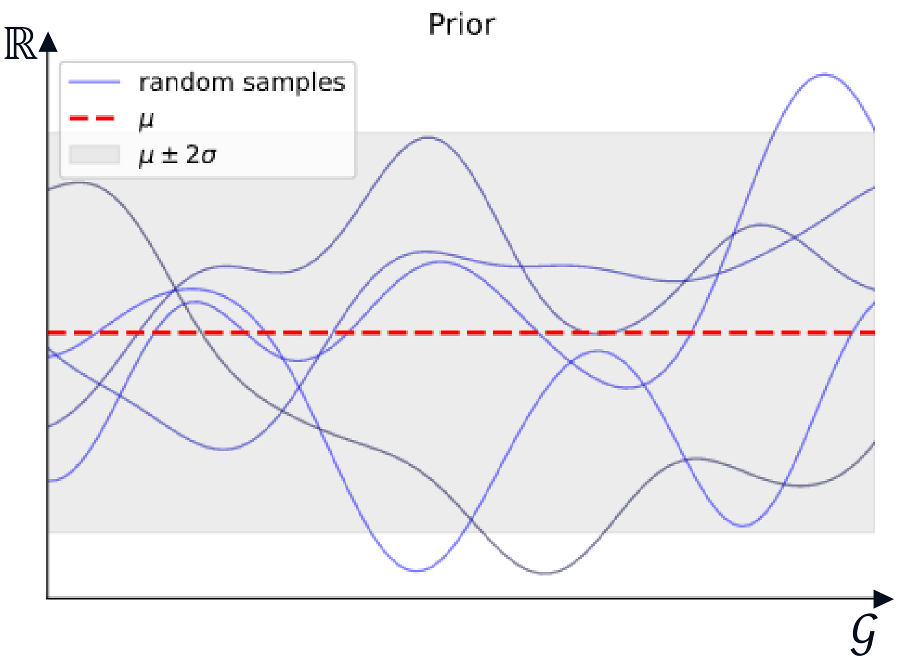
Inputs
Gaussian Process regression
Prediction
Uncertainty quantification
Pressure prediction
2.451e+04 80607 1.4e+5 1.9e+5 2.489e+05
Standard deviation pressure field
1.741e-02 0.1 0.16 0.22 2.737e-01

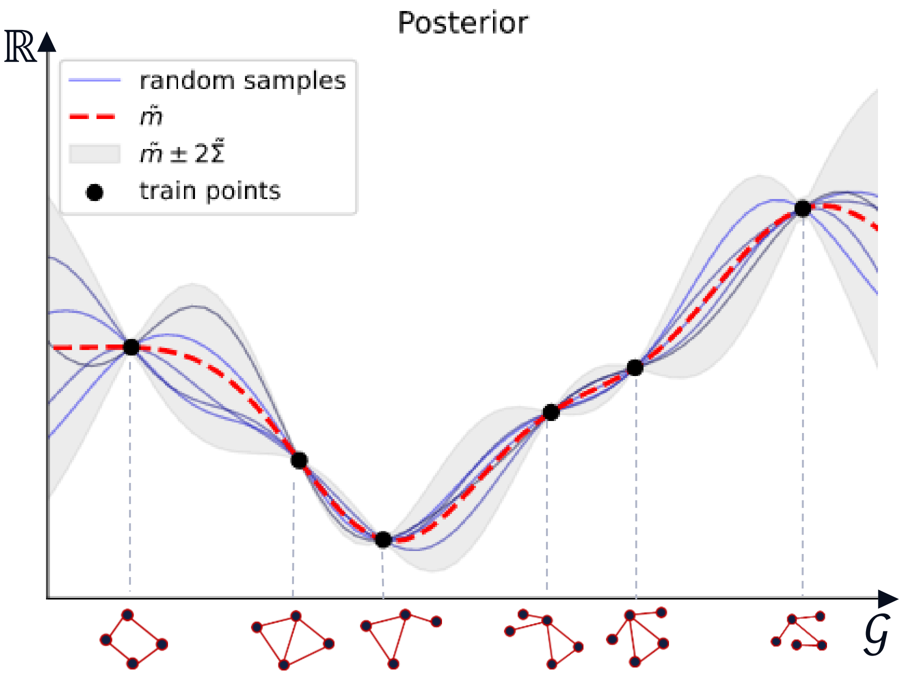
Inputs and outputs

- **Graph inputs**
  - Mesh → Graph structure
  - 3D coordinates for all nodes
- **Scalar inputs**
  - Pressure
  - Speed of rotation
- **Scalar outputs**
  - Physical quantities



Gaussian process regression





SAFRAN

Gaussian process regression

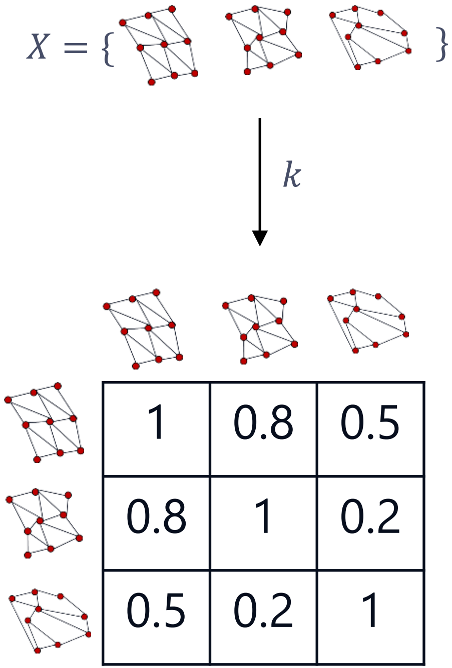
- $X = (G_1, \cdots, G_N)^T$ with $G_i \in \Gamma$ (**train input graphs**)
- $Y = (y_1, \cdots, y_N)^T$, $y_i \in \mathbb{R}$ (scalar outputs)

- Observations: $y_i = f(G_i) + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$
  $f: \Gamma \to \mathbb{R}$
- $\bar{f} = \big(f(G_1), \cdots, f(G_N)\big)^T$

- **Gaussian prior** over functions: $\bar{f} \mid G_1, \cdots, G_N \sim \mathcal{N}\big(0, K^{ff}\big)$

- K^{ff}: $N \times N$ covariance matrix where $K_{ij}^{ff} = k\big(G_i, G_j\big)$
- and $k: \Gamma \times \Gamma \to \mathbb{R}$ is a <u>**positive definite kernel**</u>

- Question: how to choose k ?

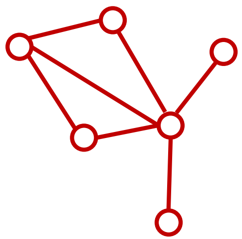


1	0.8	0.5
0.8	1	0.2
0.5	0.2	1

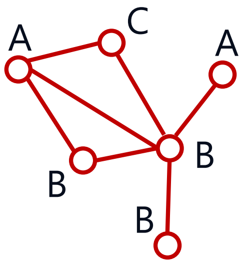
SAFRAN

What is a graph ?

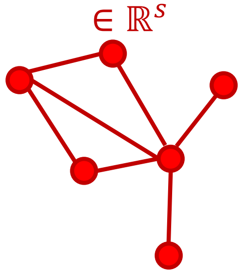
What is a graph ?



Case 1 :
Vertices + Edges

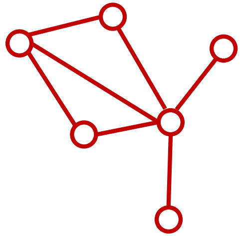


Case 2 :
Vertices + Edges
+ Node labels

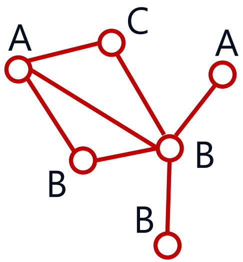


Case 3 :
Vertices + Edges
+ Node attributes

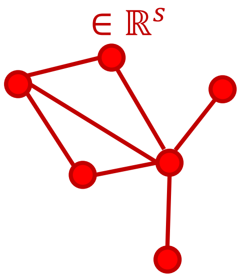
What is a graph ?



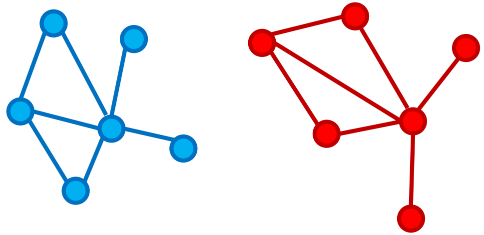
Case 1 :
Vertices + Edges



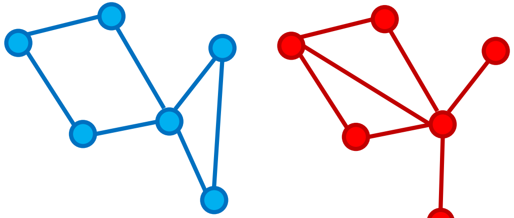
Case 2 :
Vertices + Edges
+ Node labels



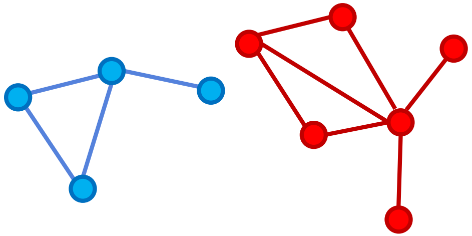
Case 3 :
Vertices + Edges
+ Node attributes



Case 3A: Fixed structure -> signal

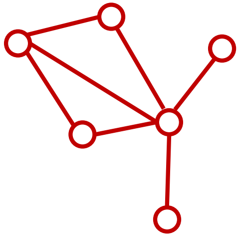


Case 3B: Fixed number of nodes

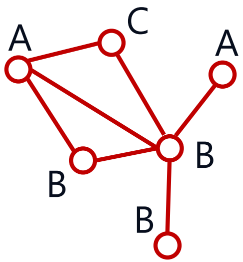


Case 3C: Varying number of nodes + structure + attributes

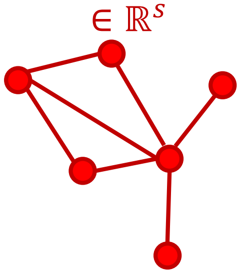
What is a graph ?



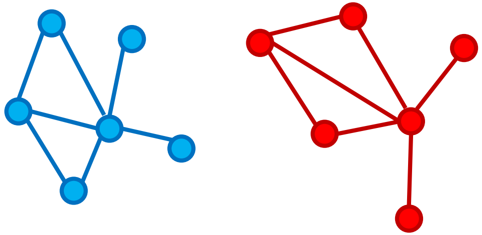
Case 1 :
Vertices + Edges



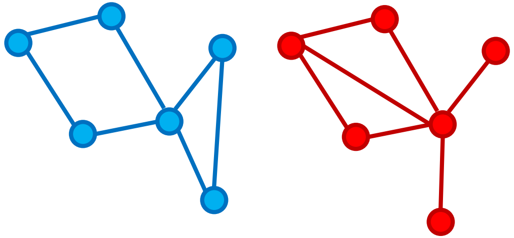
Case 2 :
Vertices + Edges
+ Node labels



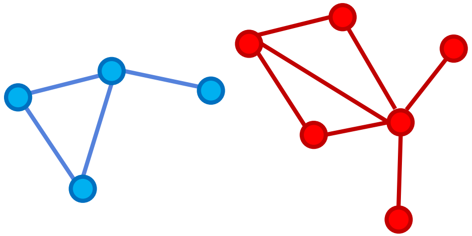
Case 3 :
Vertices + Edges
+ Node attributes



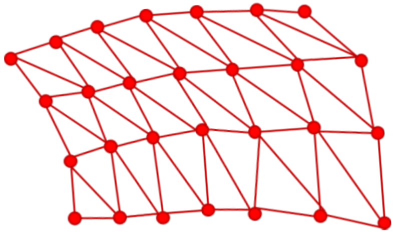
Case 3A: Fixed structure -> signal



Case 3B: Fixed number of nodes



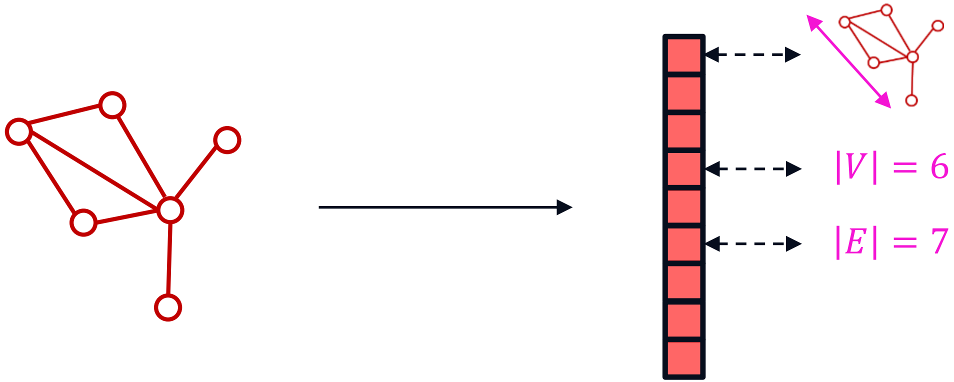
Case 3C: Varying number of
nodes + structure + attributes



Case 3C+: Varying number of
nodes + structure + attributes
+ large-scale + sparse

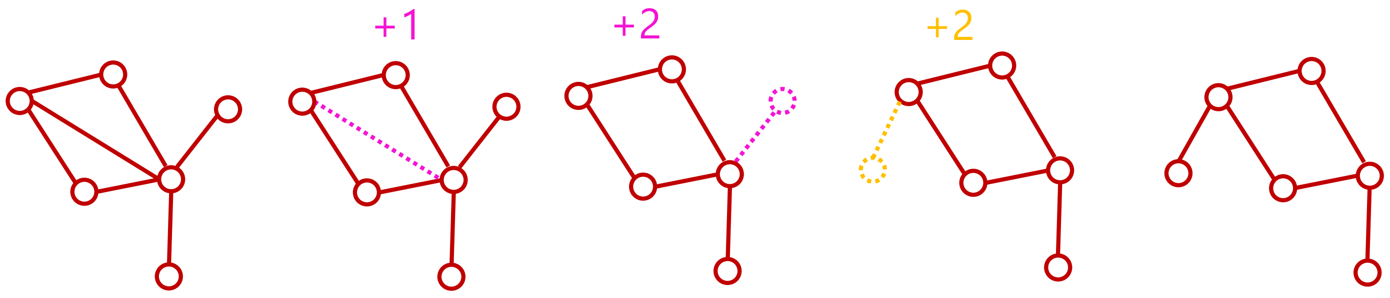
Graph kernels

Invariants / Topological descriptors



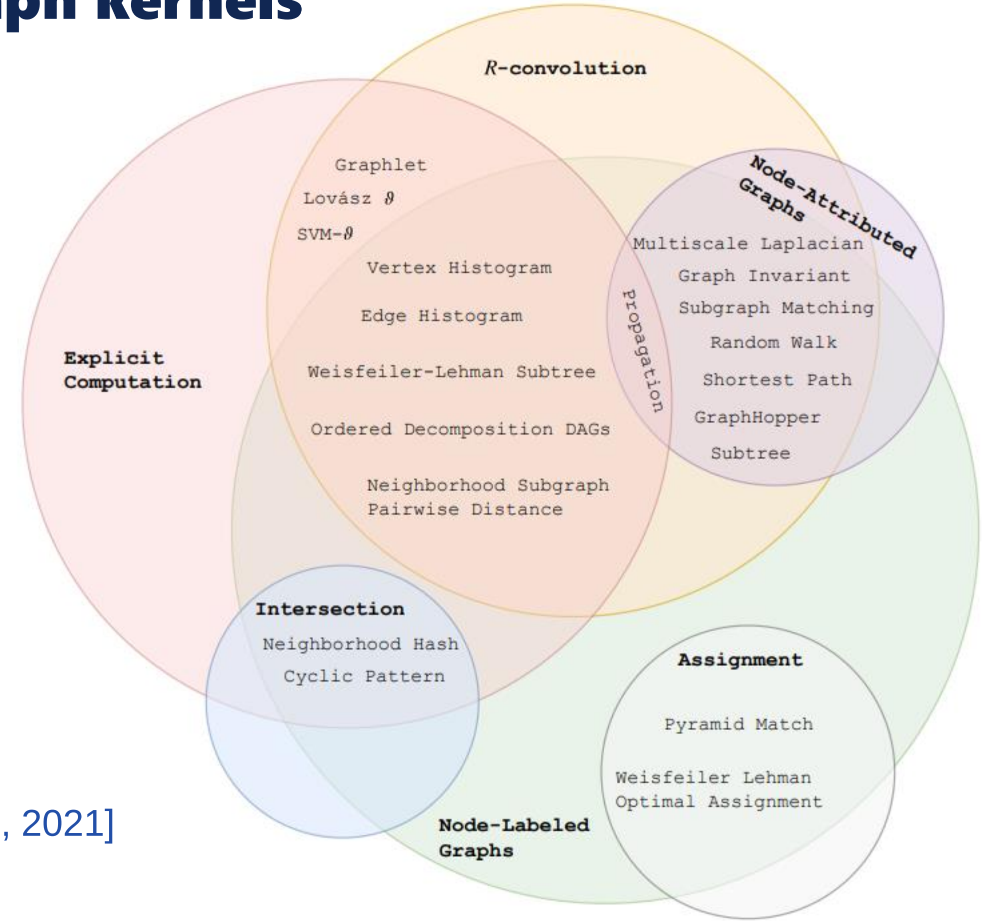
- Map the graph to a vectorial representation
- Invariants: do not change under graph isomorphism (diameter, average clustering coefficient, …)
- Complete invariants require exponential time

Graph edit distance



- $d(G_1, G_2)$ = minimal number of operations to transfrom G_1 in G_2 (adding/removing an edge/vertex, node relabeling)
- NP-complete
- Not suited for node-attributed graphs…

Taxonomy of graph kernels



Taxonomy of graph kerneks.

Figure from [Nikolentzos et al., 2021]

$\mathcal{R}$-convolution kernels

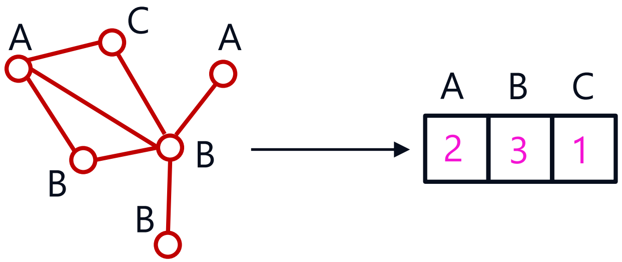
- $\mathcal{R}(g_1, \cdots, g_d, G)$: $\mathcal{R}$-decomposition where g_i is a 'part' of G (relationship)
- $\mathcal{R}^{-1}(G) = \{ g \coloneqq (g_1, \cdots, g_d) \mid \mathcal{R}(g_1, \cdots, g_d, G)\}$: pre-image of the relation
- Let k_i a base kernel based on a subset of the parts denoted G_i.
- The $\mathcal{R}$-convolution kernel between G and G' is defined as

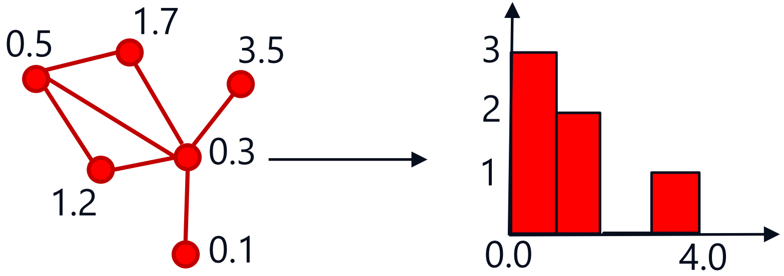
$$k_{\mathcal{R}}(G, G') \coloneqq \sum_{g \in \mathcal{R}^{-1}(G)} \sum_{g' \in \mathcal{R}^{-1}(G)} \prod_{i=1}^{d} k_i(g_i, g_i')$$

All node-pairs kernel / node histogram kernel

$$k_N(G, G') := \sum_{v \in V} \sum_{v' \in V'} k_{node}(v, v')$$ where k_{node} is a positive definite kernel between node attributes/labels -> feature map ϕ_{node}

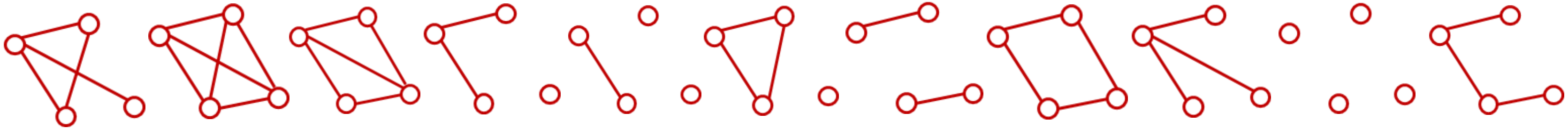
- $k_N(G, G') = \langle \phi_N(G), \phi_N(G') \rangle_{\mathcal{H}}$ where $\phi_N(G) := \sum_{v \in V} \phi_{node}(v)$
- When $\phi_{node}(v) = e_{l(v)}$ (k_{node} is a Dirac kernel on node labels), ϕ_N is an unnormalized histogram that counts occurences of node labels

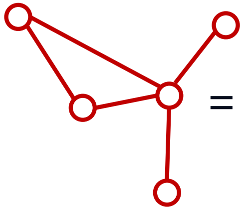




SAFRAN

Graphlet kernel





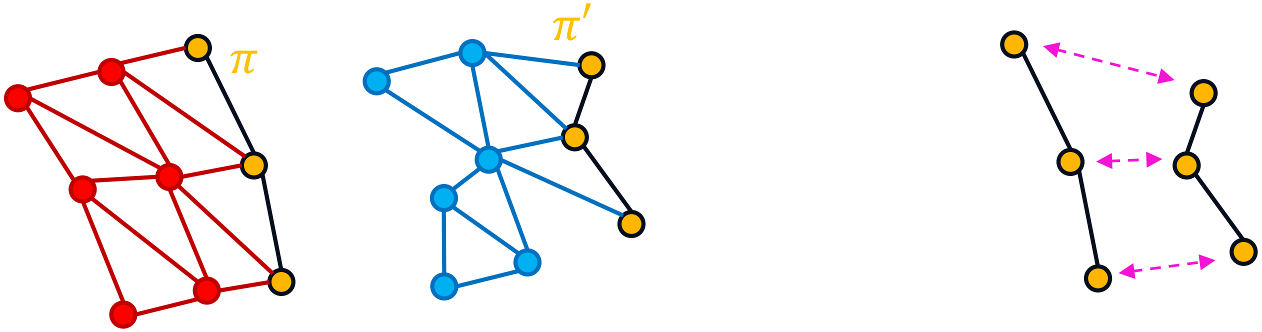
=

2	0	0	0	1	0	0	0	2	0	0

- Set of k-graphlets of size N_k, $k \geq 3$
- k-spectrum of G: vector $\phi_{GL}(G)$ of the frequencies of all graphlets in G
- $k_{GL}(G, G') \coloneqq \phi_{GL}(G)\phi_{GL}(G')^T$
- Issue: does not take into account labels or attributes

Graph Hopper

[Feragen et al., 2013]

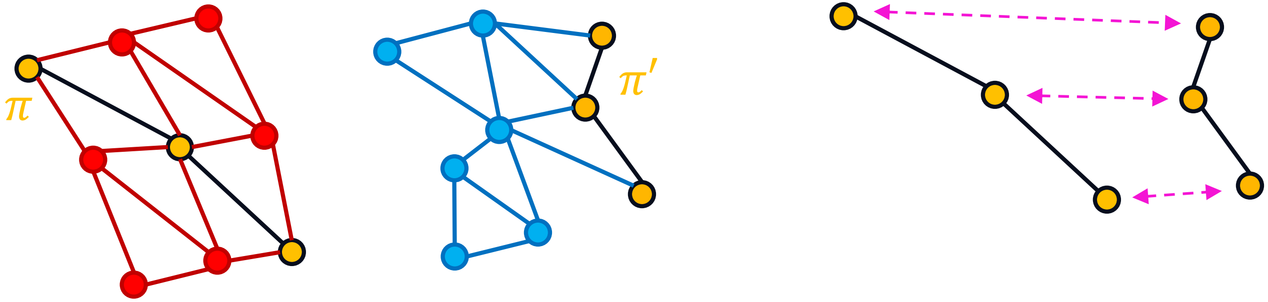


$$k(G,G') \coloneqq \sum_{\pi\in\mathcal{P},\pi'\in\mathcal{P}'} k_p(\pi,\pi') \;\; with\; k_p(\pi,\pi') \coloneqq \begin{cases} \sum_{j=1}^{|\pi|} RBF(\pi_j,\pi'_j) \;\; if\; |\pi| = |\pi'| \\ 0 \qquad\qquad\qquad\qquad otherwise \end{cases}$$

- $\mathcal{P}$: set of all shortest paths in G, $|\pi|$: discrete length of the path $\pi = (\pi_1, \cdots, \pi_{|\pi|})$
- Complexity: $O(n^2(|E| + \log n))$

Graph Hopper

[Feragen et al., 2013]



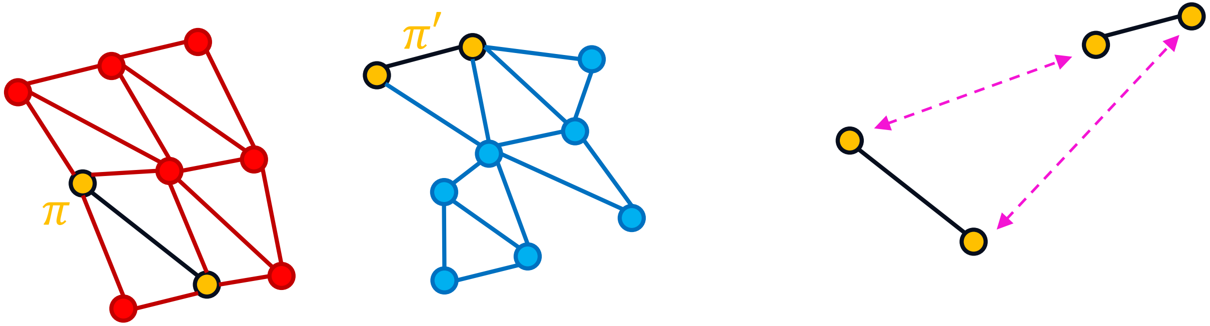
$$k(G,G') \coloneqq \sum_{\pi\in\mathcal{P},\pi'\in\mathcal{P}'} k_p(\pi,\pi') \;\; with\; k_p(\pi,\pi') \coloneqq \begin{cases} \sum_{j=1}^{|\pi|} RBF(\pi_j,\pi'_j) \;\; if\; |\pi| = |\pi'| \\ 0 \qquad\qquad\qquad\qquad otherwise \end{cases}$$

- $\mathcal{P}$: set of all shortest paths in G, $|\pi|$: discrete length of the path $\pi = (\pi_1, \cdots, \pi_{|\pi|})$
- Complexity: $O(n^2(|E| + \log n))$

SAFRAN

Graph Hopper

[Feragen et al., 2013]



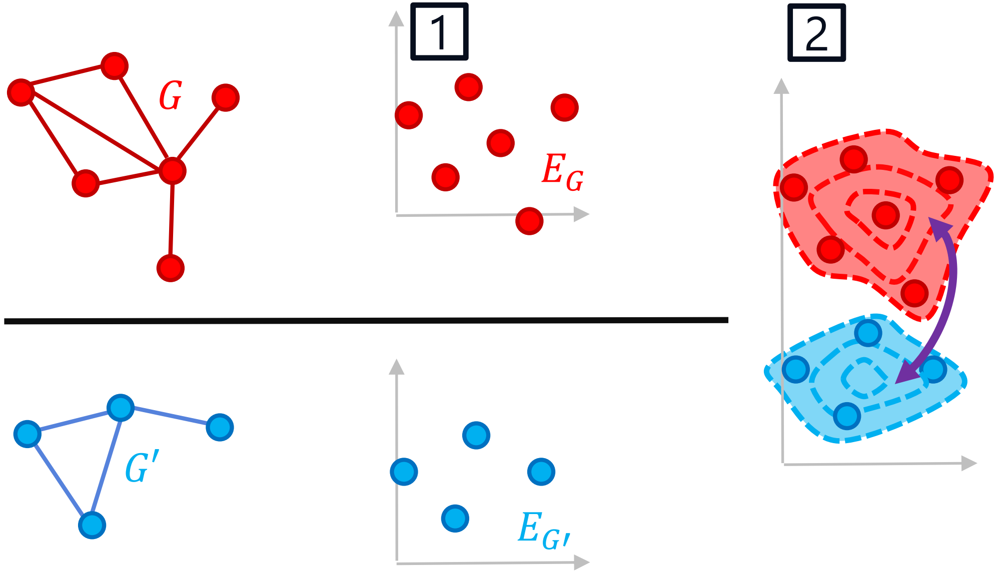
$$k(G, G') \coloneqq \sum_{\pi \in \mathcal{P}, \pi' \in \mathcal{P}'} k_p(\pi, \pi') \;\; with \; k_p(\pi, \pi') \coloneqq \begin{cases} \sum_{j=1}^{|\pi|} RBF(\pi_j, \pi'_j) \;\; if \; |\pi| = |\pi'| \\ 0 \qquad\qquad\qquad\qquad otherwise \end{cases}$$

- $\mathcal{P}$: set of all shortest paths in G, $|\pi|$: discrete length of the path $\pi = (\pi_1, \cdots, \pi_{|\pi|})$
- Complexity: $O(n^2(|E| + \log n))$

SAFRAN

Sliced Wasserstein Weisfeiler Lehman (SWWL)

Node embeddings + Optimal transport approaches

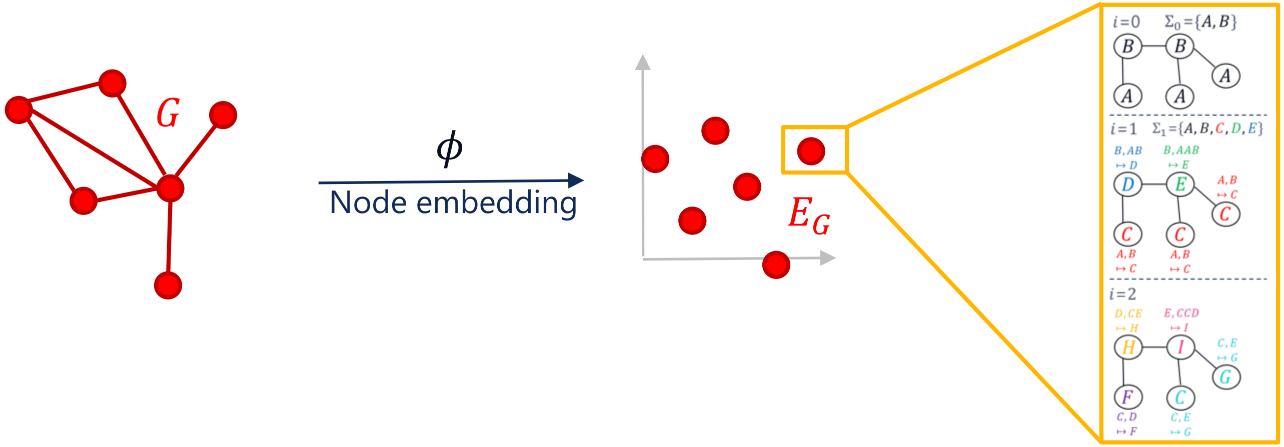


SAFRAN

Wasserstein Weisfeiler-Lehman Graph kernel (step 1)

[Togninalli et al., 2019]

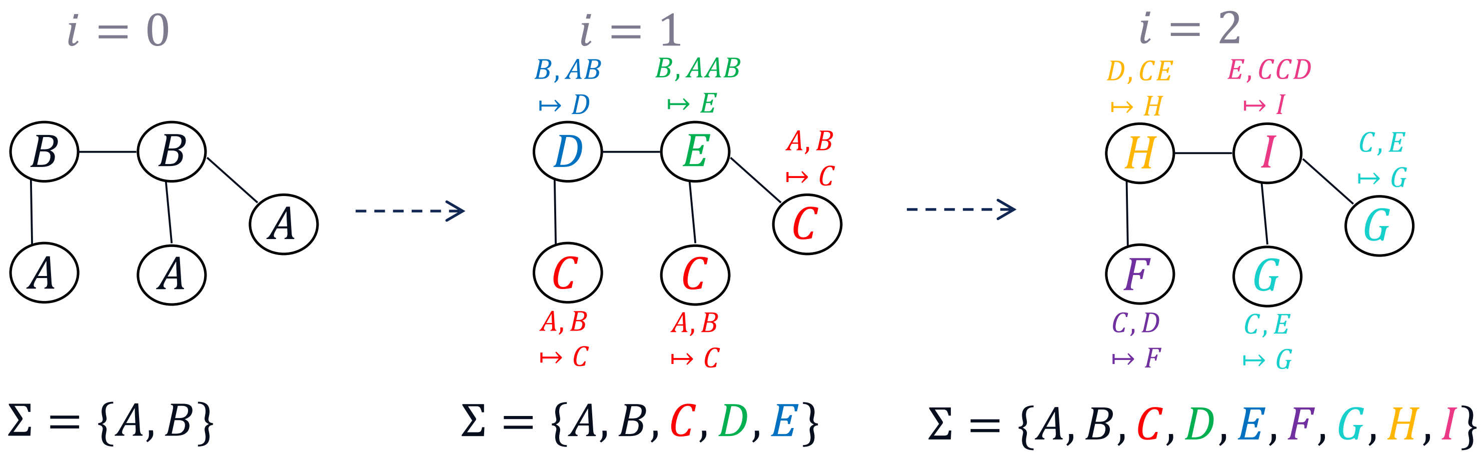
$\boxed{1}$



Weisfeiler-Lehman embeddings

Figure From [Kriege et al., 2020]

- WL relabeling (discrete case)



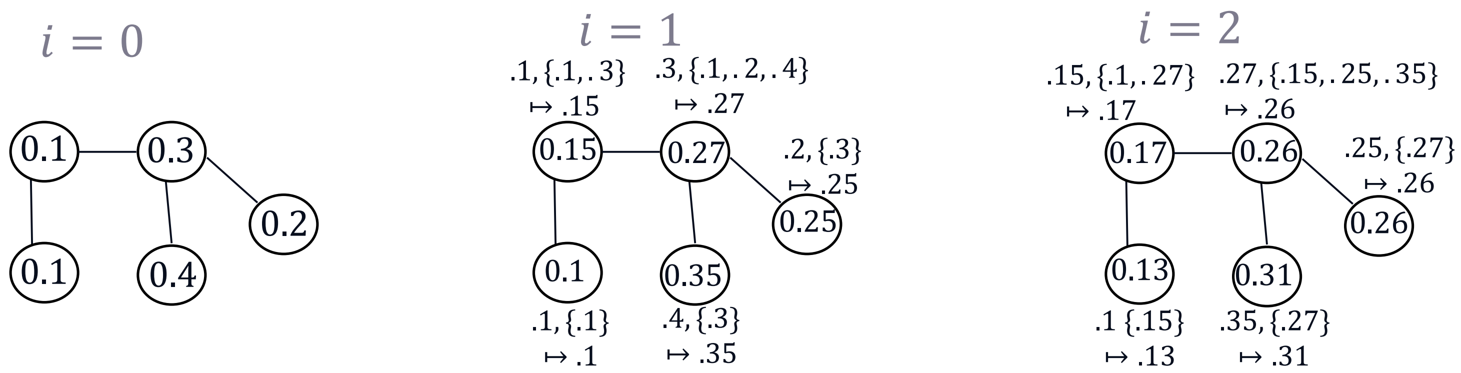
$$l^{(i+1)}(v) = Hash\left(l^i(v), \{l^i(u), u \in \mathcal{N}(v)\}\right)$$

$$X_G^{(i)} = \left[l^{(i)}(v), v \in V_G\right] \qquad X_G = Concatenate(X_G^{(0)}, \cdots, X_G^{(H)})$$

Continuous Weisfeiler-Lehman embeddings

[Togninalli et al., 2019]

- WL relabeling (continuous case)

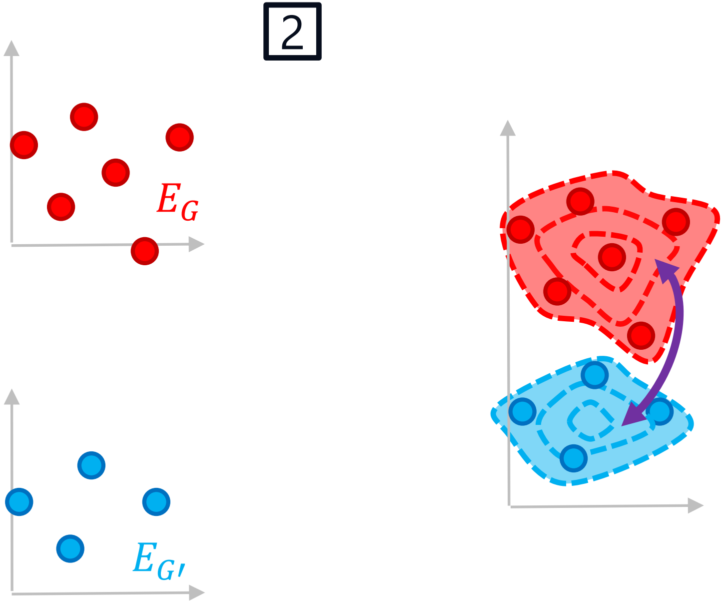


$$a^{(i+1)}(v) = \frac{1}{2}\left(a^{(i)}(v) + \frac{1}{\deg(v)} \sum_{u \in \mathcal{N}(v)} w(v,u)\, a^{(i)}(u)\right)$$

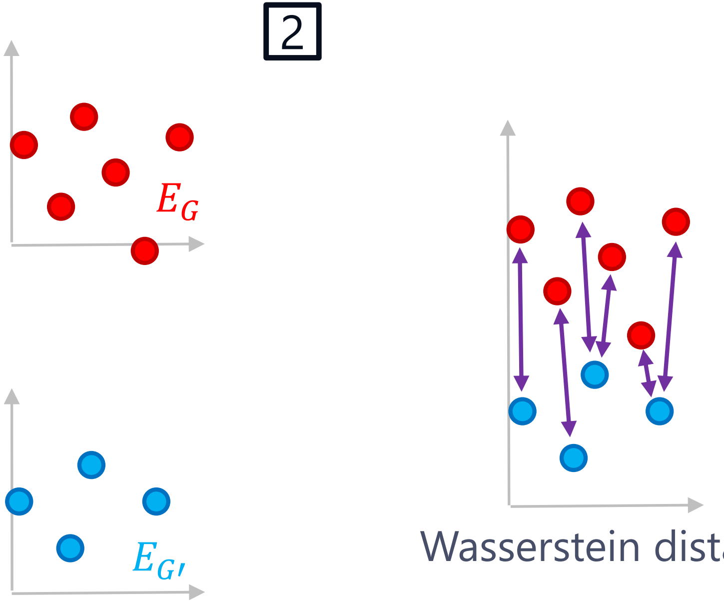
$$X_G^{(i)} = \left[a^{(i)}(v), v \in V_G\right] \qquad X_G = Concatenate\left(X_G^{(0)}, \cdots, X_G^{(H)}\right)$$

SAFRAN

Wasserstein Weisfeiler-Lehman graph kernel (step 2)



Wasserstein Weisfeiler-Lehman graph kernel (step 2)



Wasserstein distance

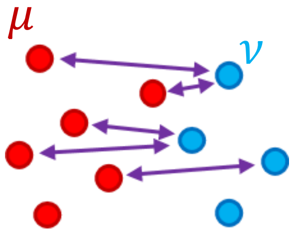
Wasserstein distance

- $\forall r \in [1, +\infty)$, $\mathcal{P}_r(\mathbb{R}^s)$: probability measures on $\mathbb{R}^s$ with finite moments of order r.

$$\forall \mu, \nu \in \mathcal{P}_r(\mathbb{R}^s), \mathcal{W}_r^r(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathbb{R}^s \times \mathbb{R}^s} \|x - y\|^r d\pi(x, y)$$

where:
- $\|.\|$ denotes the Euclidean norm,
- $\Pi(\mu, \nu)$ the set of probability measures on $\mathbb{R}^s \times \mathbb{R}^s$ whose marginals w.r.t. the 1st/2nd variable are resp. μ and ν

- Discrete case: $\mu = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$ $\qquad \nu = \frac{1}{n'}\sum_{i=1}^{n'} \delta_{y_i}$



SAFRAN

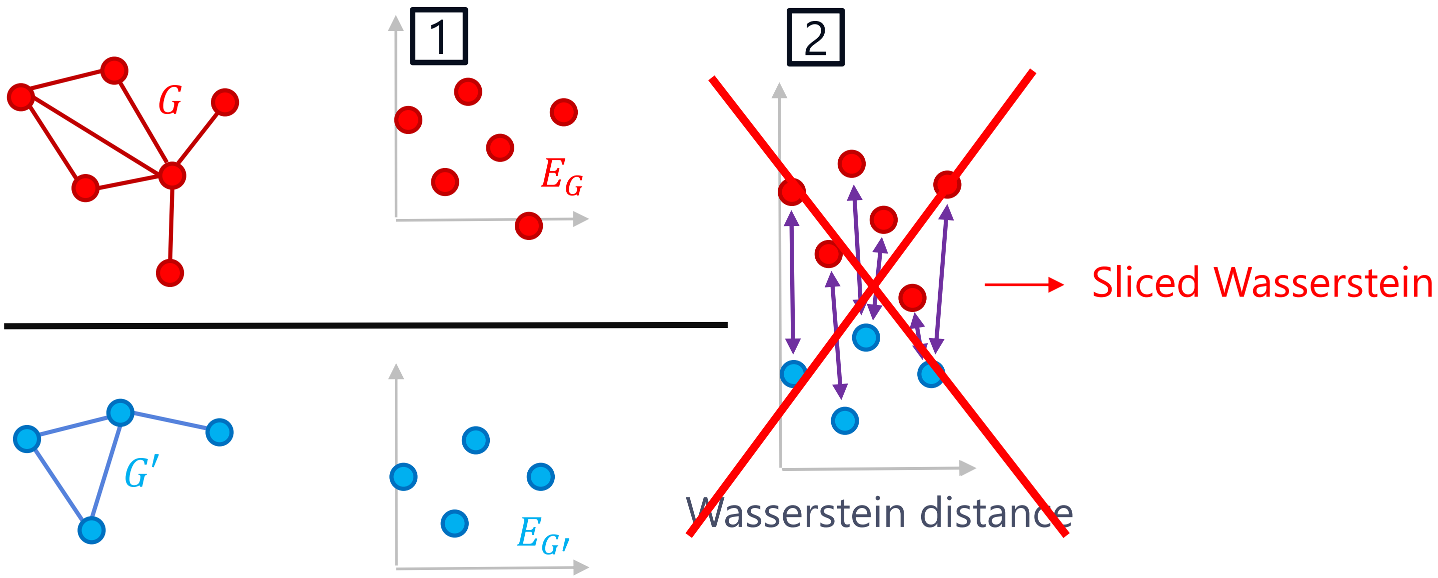
Wasserstein distance: issues

- ✗ Impossible to build a positive definite kernel (*in dimension ≥ 2 *) [Peyré, Cuturi, 2019]
- ✗ Computationally expensive : $O(n^3 \log(n))$

- Use case: 1000 graphs with 30 000 vertices
  → **400 days** to build the Gram matrix...

Sliced Wasserstein Weisfeiler Lehman graph kernel

[Us]



Idea: replace Wasserstein by **sliced Wasserstein** !

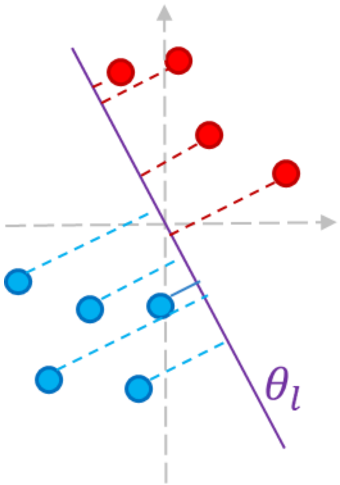
→ ✓ $\mathbf{O}(\boldsymbol{n}\,\boldsymbol{log}(\boldsymbol{n}))$ and ✓ **positive definite** substitution kernels

[Meunier et al., 2022]

Sliced Wasserstein distance

- The **sliced Wasserstein** distance is defined as:

$$\mathcal{SW}_r^r(\mu,\nu) = \int_{\mathbb{S}^{s-1}} \mathcal{W}_r^r(\theta_\#^*\mu, \theta_\nu^*) \mathrm{d}\sigma(\theta)$$



where

- $\mathbb{S}^d$: d-dimensional unit sphere, σ : uniform distribution on $\mathbb{S}^d$
- $\theta_\#^*\mu$: push-forward measure of $\mu \in \mathcal{P}_r(\mathbb{R}^s)$ by $\theta^* \begin{pmatrix} \mathbb{R}^s \to \mathbb{R} \\ x \mapsto \langle\theta, x\rangle \end{pmatrix}$

$$\mathcal{W}_r^r(\mu,\nu) = \int_0^1 |\mathrm{F}^{-1}(\mu) - F^{-1}(\nu)|^{\mathrm{r}}\, \mathrm{d}t$$

1-d Wasserstein distances between

μ and ν

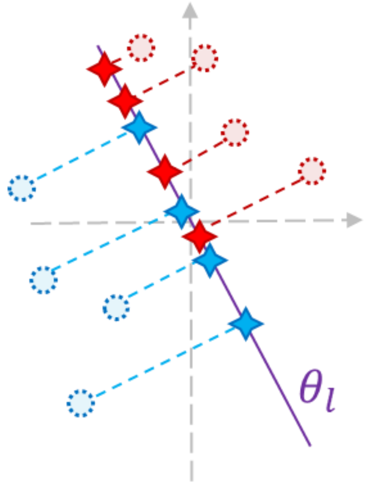
Quantile function

SAFRAN

Sliced Wasserstein distance

- The **sliced Wasserstein** distance is defined as:

$$\mathcal{SW}_r^r(\mu,\nu) = \int_{\mathbb{S}^{s-1}} \mathcal{W}_r^r(\theta_{\#}^*\mu, \theta_\nu^*)\mathrm{d}\sigma(\theta)$$



where

- $\mathbb{S}^d$: d-dimensional unit sphere, σ : uniform distribution on $\mathbb{S}^d$
- $\theta_{\#}^*\mu$: push-forward measure of $\mu \in \mathcal{P}_r(\mathbb{R}^s)$ by $\theta^* \begin{pmatrix} \mathbb{R}^s \to \mathbb{R} \\ x \mapsto \langle\theta, x\rangle \end{pmatrix}$

$$\boxed{\mathcal{W}_r^r(\mu,\nu) = \frac{1}{n}\sum_{i=1}^{n} |x_{(i)} - y_{(i)}|^r}$$

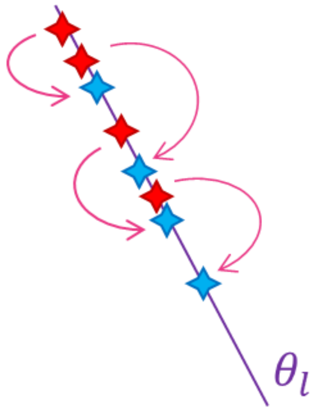
1-d Wasserstein distances between

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i} \text{ and } \nu = \frac{1}{n}\sum_{i=1}^{n}\delta_{y_i}$$

Sliced Wasserstein distance

- The **sliced Wasserstein** distance is defined as:

$$\mathcal{SW}_r^r(\mu,\nu) = \int_{\mathbb{S}^{s-1}} \mathcal{W}_r^r(\theta_\#^*\mu, \theta_\nu^*)\mathrm{d}\sigma(\theta)$$



where

- $\mathbb{S}^d$: d-dimensional unit sphere, σ : uniform distribution on $\mathbb{S}^d$
- $\theta_\#^*\mu$: push-forward measure of $\mu \in \mathcal{P}_r(\mathbb{R}^s)$ by $\theta^* \begin{pmatrix} \mathbb{R}^s \to \mathbb{R} \\ x \mapsto \langle\theta, x\rangle \end{pmatrix}$

$$\boxed{\mathcal{W}_r^r(\mu,\nu) = \frac{1}{n}\sum_{i=1}^{n} |x_{(i)} - y_{(i)}|^r}$$

1-d Wasserstein distances between

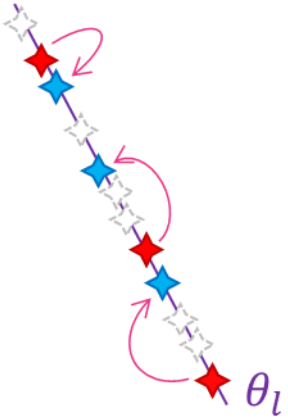
$$\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i} \text{ and } \nu = \frac{1}{n}\sum_{i=1}^{n}\delta_{y_i}$$

SAFRAN

Sliced Wasserstein distance

- The **sliced Wasserstein** distance is defined as:

$$\mathcal{SW}_r^r(\mu,\nu) = \int_{\mathbb{S}^{s-1}} \mathcal{W}_r^r(\theta_{\#}^*\mu, \theta_{\nu}^*)\mathrm{d}\sigma(\theta)$$



where

- $\mathbb{S}^d$: d-dimensional unit sphere, σ : uniform distribution on $\mathbb{S}^d$
- $\theta_{\#}^*\mu$: push-forward measure of $\mu \in \mathcal{P}_r(\mathbb{R}^s)$ by $\theta^* \begin{pmatrix} \mathbb{R}^s \to \mathbb{R} \\ x \mapsto \langle\theta, x\rangle \end{pmatrix}$

$$\boxed{\widehat{\mathcal{W}_r^r}(\mu,\nu) = \frac{1}{Q}\sum_{q=1}^{Q} \left|x_{(q)} - y_{(q)}\right|^r}$$

(Approximation with $Q \ll \max(n, n')$ quantiles)

1-d Wasserstein distances between

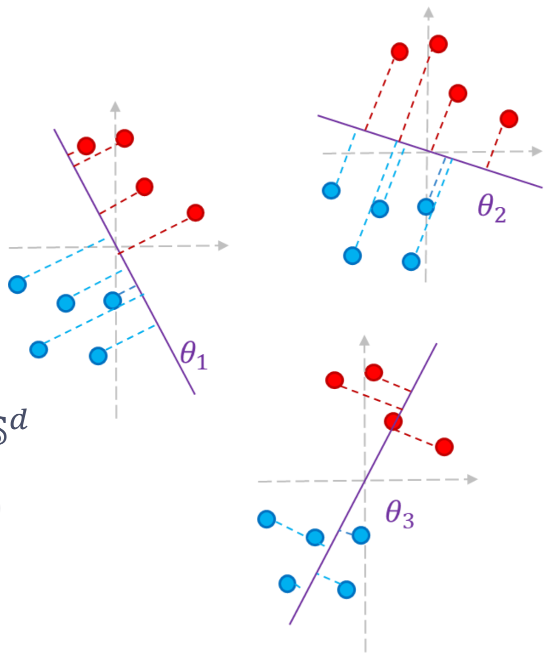
$$\mu = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i} \text{ and } \nu = \frac{1}{n'}\sum_{i=1}^{n'} \delta_{y_i}$$

SAFRAN

Sliced Wasserstein distance

- The (estimated) **sliced Wasserstein** distance is defined as:

$$\boxed{\widehat{\mathcal{SW}_r^r}(\mu,\nu) = \frac{1}{P}\sum_{p=1}^{P} \widehat{\mathcal{W}_r^r}\left(\left(\theta_p^*\right)_{\#}\mu, \left(\theta_p^*\right)_{\#}\nu\right)}$$



where

- $\mathbb{S}^d$: d-dimensional unit sphere, σ : uniform distribution on $\mathbb{S}^d$
- $\theta_{\#}^*\mu$: push-forward measure of $\mu \in \mathcal{P}_r(\mathbb{R}^s)$ by $\theta^* \begin{pmatrix} \mathbb{R}^s \to \mathbb{R} \\ x \mapsto \langle \theta, x \rangle \end{pmatrix}$

$$\widehat{\mathcal{W}_r^r}(\mu,\nu) = \frac{1}{Q}\sum_{q=1}^{Q} \left|x_{(q)} - y_{(q)}\right|^r$$

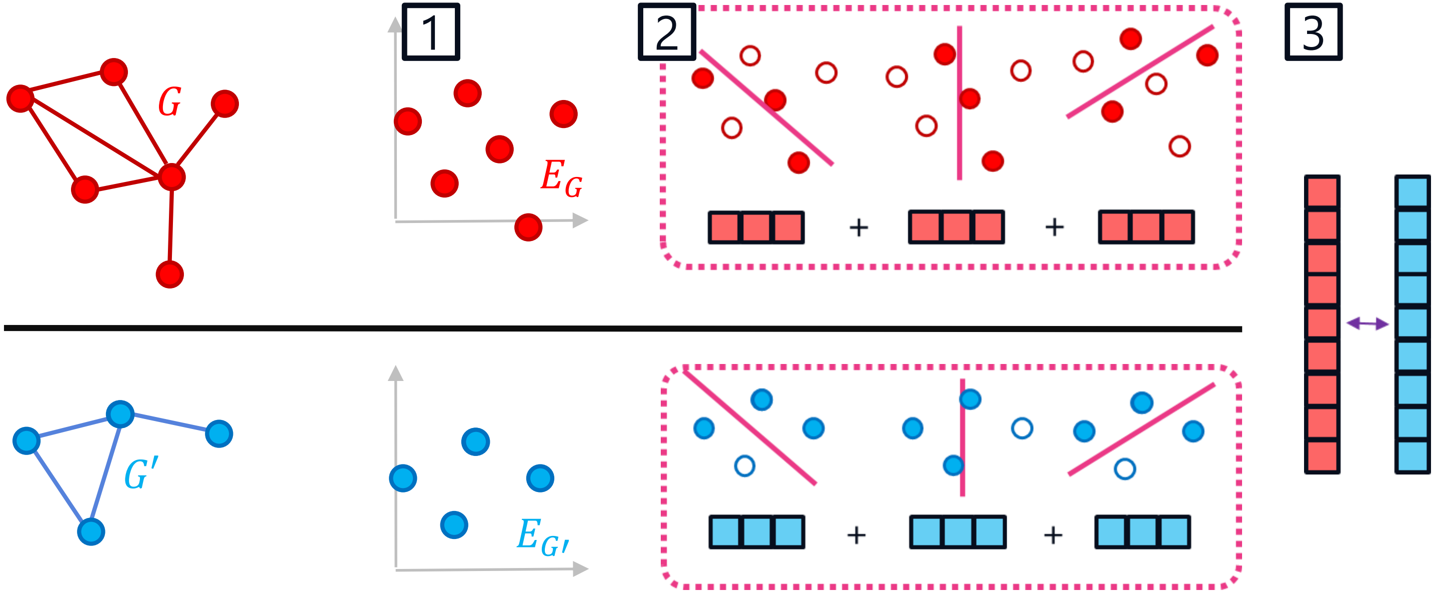
(Approximation with $Q \ll \max(n, n')$ quantiles)

1-d Wasserstein distances between

$$\mu = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i} \text{ and } \nu = \frac{1}{n'}\sum_{i=1}^{n'} \delta_{y_i}$$

Sliced Wasserstein Weisfeiler Lehman (SWWL)

[Us]



Sliced Wasserstein Weisfeiler Lehman (SWWL)

[Us]

- $\phi: G \mapsto X_G \in \mathbb{R}^{|V_G| \times d(H+1)}$: WL embeddings after H iterations

- $k_{SWWL}(G, G') = e^{-\lambda \widehat{\mathcal{SW}_2^2}(\phi(G),\, \phi(G'))}$ (* considering by abuse $\phi(G), \phi(G')$ as empirical measures *)

with
$$\widehat{\mathcal{SW}_2^2}(\mu, \nu) = \frac{1}{\mathrm{PQ}} \sum_{\mathrm{p}=1}^{\mathrm{P}} \sum_{q=1}^{Q} \left| u_q^{\theta_p} - u'^{\theta}_q \right|^2 = \left\| E_{\phi(G)} - E_{\phi(G')} \right\|_2^2$$

$\rightarrow$ Precomputed embeddings $E_{\phi(G)}, E_{\phi(G')} \in \mathbb{R}^{PQ}$ where $u_q^{\theta_p} = \langle \theta_p, \phi(G) \rangle_{(q)}$

$$E_{\phi(G)} = [u_1^{\theta_1}, \cdots, u_Q^{\theta_1}, \cdots, u_1^{\theta_P}, \cdots, u_Q^{\theta_P}]$$

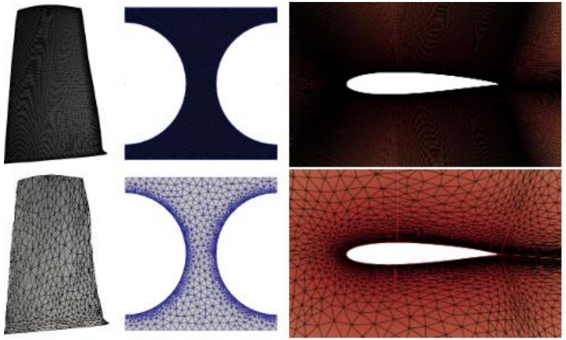
- Complexity for the Gram matrix (sparse graphs):

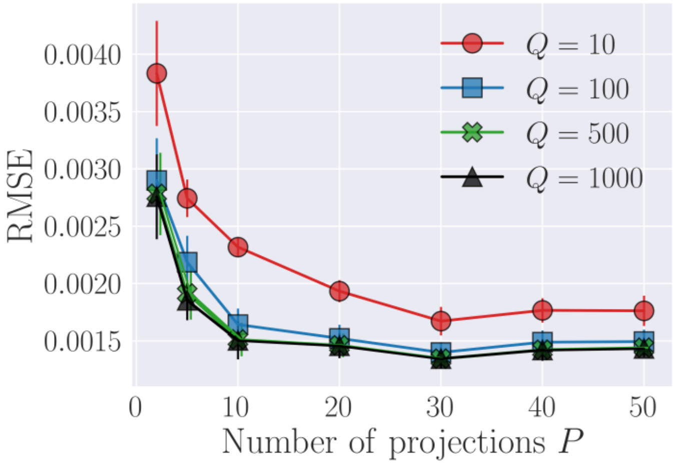
$$O(\underbrace{\boxed{NHn}}_{\text{WL iterations}} + \underbrace{\boxed{NP\, n \log n}}_{\text{Quantiles}} + \underbrace{\boxed{N^2 PQ}}_{\text{Usual RBF kernel}})$$

SAFRAN

SWWL: experiments on meshes

[Us]





RMSE (5 exp)

Kernel/Dataset	Rotor37 x10^{-3}	Rotor37-CM x10^{-3}	Tensile2d x1	Tensile2d-CM x1	AirfRANS x10^{-4}	AirfRANS-CM x10^{-4}
SWWL	1.44 ± 0.07	$\mathbf{3.49 \pm 0.15}$	$\mathbf{0.89 \pm 0.01}$	$\mathbf{1.51 \pm 0.01}$	7.56 ± 0.36	9.63 ± 0.54
WWL	-	$\mathbf{3.51 \pm 0.00}$	-	6.46 ± 0.00	-	14.4 ± 0.80
PK	-	4.18 ± 0.39	-	6.03 ± 4.58	-	8.94 ± 2.31

Time to build the Gram matrix

Kernel/Dataset	Rotor37	Rotor37-CM	Tensile2d	Tensile2d-CM	AirfRANS	AirfRANS-CM
SWWL	**1min + 11s**	**4s + 11s**	**11s + 4s**	**2s + 4s**	**5min + 7s**	**15s + 7s**
WWL	-	13min (*)	-	6min (*)	-	8h (*)
PK	-	1min	-	2min	-	15min

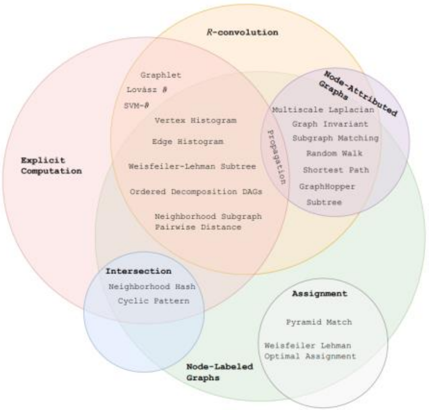
(*) in parallel, using 100 jobs

SAFRAN

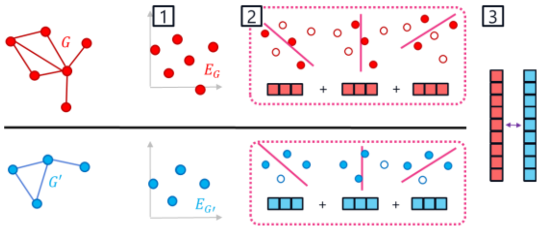
Conclusion and future work

Conclusion

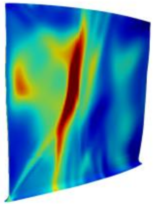
- Limits of existing graph kernels
  - Many do not handle continuous attributes
  - Many do not scale well to large graphs
  - Many do not guarantee positive definiteness
  - Many are too dependent on the graph structure



- We propose the Sliced Wasserstein Weisfeiler Lehman (SWWL) kernel
  - Positive definite
  - Tractable for large graphs
  - Competitive results for mesh-based Gaussian process regression



- Future work
  - Extension to multiple outputs (e.g. vector fields)



SAFRAN

References

- **Graph kernels, Gaussian processes**

  - Nikolentzos, G., Siglidis, G., & Vazirgiannis, M. (2021). Graph kernels: A survey. Journal of Artificial Intelligence Research, 72, 943-1027.
  - Kriege, N. M., Johansson, F. D., & Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, *5*(1), 1-42.
  - Feragen, A., Kasenburg, N., Petersen, J., de Bruijne, M., & Borgwardt, K. (2013). Scalable kernels for graphs with continuous attributes. Advances in neural information processing systems, 26.
  - Williams, C. K., & Rasmussen, C. E. (2006). Gaussian processes for machine learning (Vol. 2, No. 3, p. 4). Cambridge, MA: MIT press.

- **Optimal transport**

  - Peyré, G., & Cuturi, M. (2019). Computational optimal transport: With applications to data science. Foundations and Trends® in Machine Learning, 11(5-6), 355-607.
  - Togninalli, M., Ghisu, E., Llinares-López, F., Rieck, B., & Borgwardt, K. (2019). Wasserstein weisfeiler-lehman graph kernels. *Advances in neural information processing systems*, *32*.
  - Meunier, D., Pontil, M., & Ciliberto, C. (2022, June). Distribution Regression with Sliced Wasserstein Kernels. In International Conference on Machine Learning (pp. 15501-15523). PMLR.

Acknowledgments

- This work was supported by the French National Research Agency (ANR) through the SAMOURAI project under grant ANR20-CE46-0013.

Other approaches using Optimal Transport

Other approaches

- Many approaches with **GCNN**s and **message passing layers**
→ Continuous WL of torch_geometric

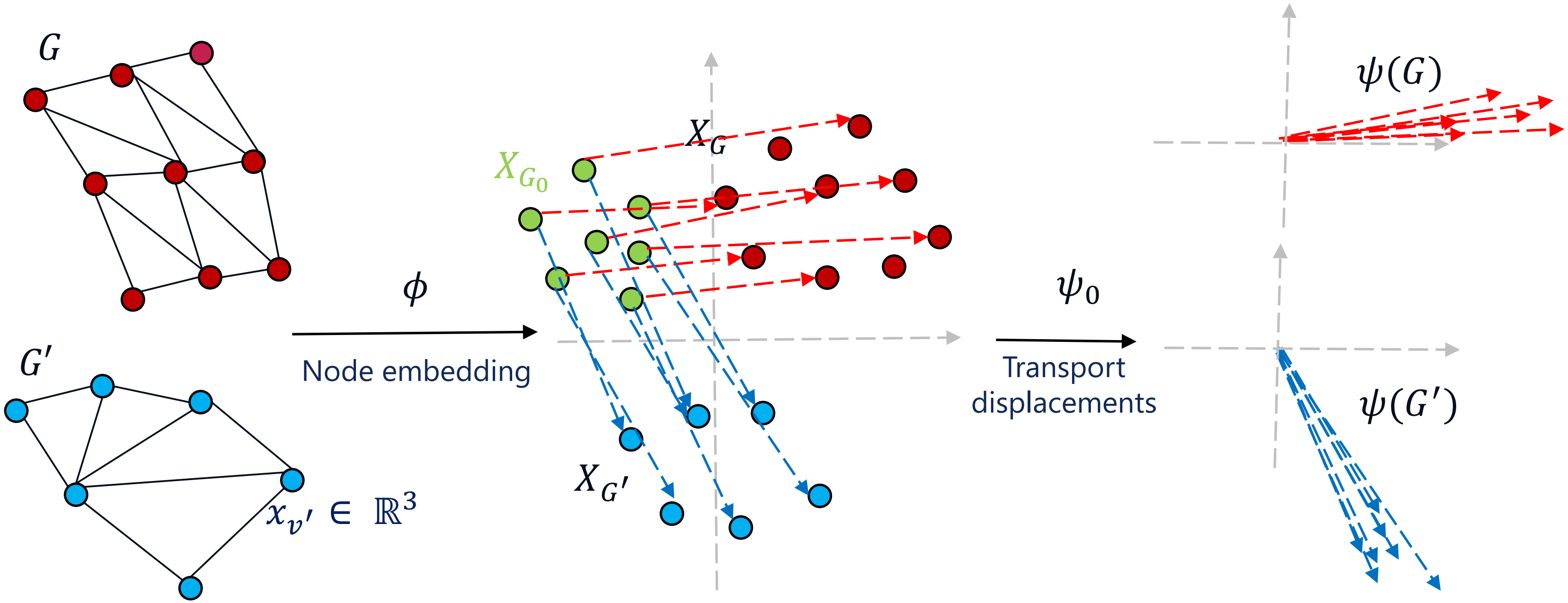
- Other node embedding: $a^{(i+1)}(v) = \sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{w(v,u)}{\sqrt{\deg(u) \deg(v)}} a^{(i)}(u)$

- Wasserstein embeddings with Linear Optimal transport [Kolouri et al., 2020]
- Pooling by Sliced-Wasserstein (PSWE) [Naderializadeh., 2021]

- Template-based GNN with OT [Vincent-Cuaz et al., 2022]

Wasserstein embeddings

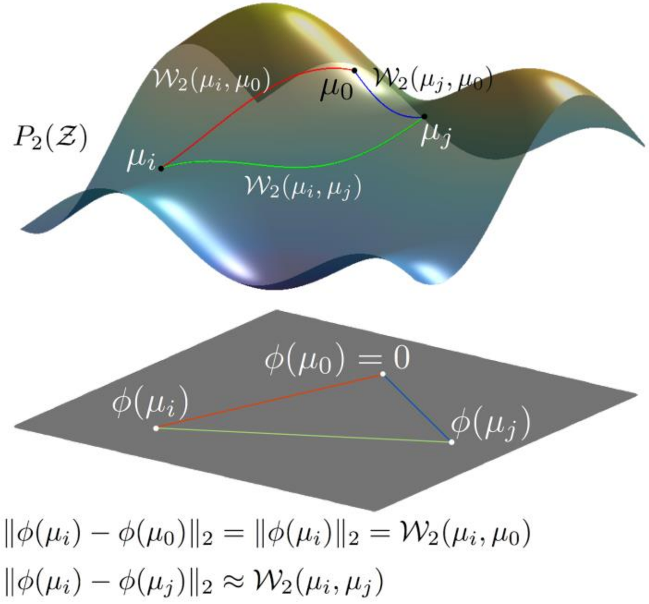
[Kolouri et al., 2020]



- Linear Wasserstein embedding (Linear Optimal transport LOT Framework)
- Transport displacements from a reference distribution to node embeddings

Wasserstein embeddings

[Kolouri et al., 2020]



- Given a first node embedding $\phi: G \mapsto X_G \in \mathbb{R}^{|V_G| \times s}$

- $X_0 \in \mathbb{R}^{n_0 \times s}$ reference node embedding
- Linear Wasserstein embedding:
- $\psi_0(X_G) \coloneqq \left(u_{G,0} - Id\right)\sqrt{n_0}$
- where $u_{G,0}$ is the Monge map that pushes X_0 to X_G

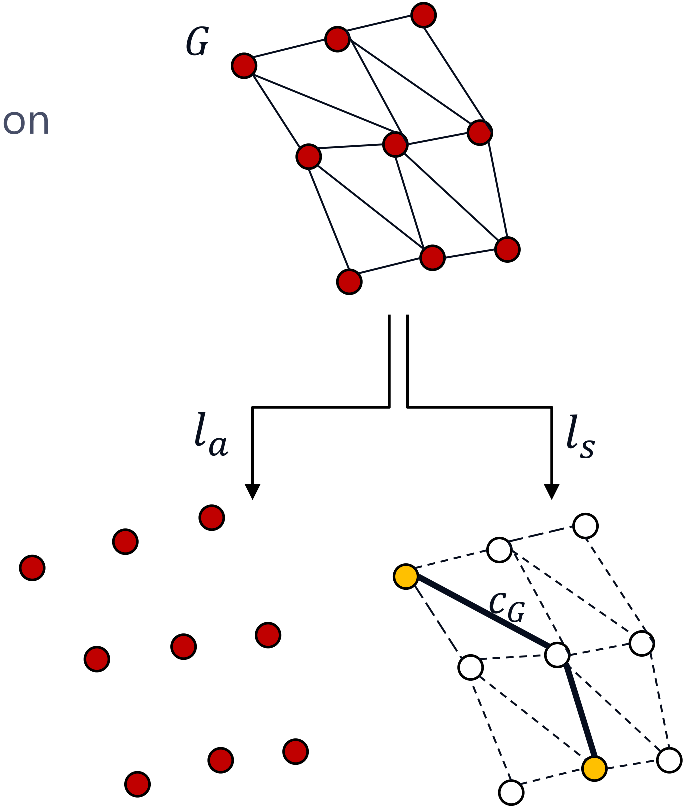
- New graph embedding: $\psi(G) \coloneqq \psi_0\big(\phi(G)\big) \in \mathbb{R}^{n_0 \times s}$ of fixed size
- **Only N Monge map calculations needed**

- Choice of the reference embedding? (Not clear)

Fused Gromov-Wasserstein distance

[Vayer et al., 2019]

- $G = (V_G, E_G, l_a, l_s)$ with $l_a : V_G \to \mathbb{R}^3$ the coordinate function
- $l_s : V_G \to \Omega_G$ with (Ω_G, c_G) a metric space dependant of G
- $c_G : \Omega_G \times \Omega_G \to \mathbb{R}_+$ 'similarity' of points **in** G (structure-dependent)
  e.g. : $c_G(l_s(v_1), l_s(v_2)) = d_{PCC}(v_1, v_2|G)$
- $a_i = l_a(v_i)$, $s_i = l_s(v_i)$: attributes/structure of point i
- $\mu_G = \sum_{i=1}^{n_G} \frac{1}{n_G} \delta_{(a_i, s_i)}$: measure of G
- $C_G = \left[c_G(s_i, s_j)\right]_{1 \le i,j \le n_G}$, $C_{G'} = \left[C_{G'}(s'_i, s'_j)\right]_{1 \le i,j \le n_{G'}}$



Fused Gromov-Wasserstein distance

[Vayer et al., 2019]

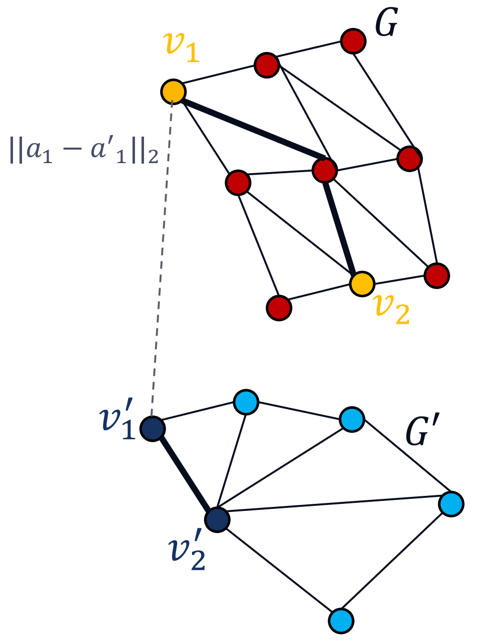
- $L_{G,G'} = |C_G[i,k] - C_{G'}[j,l]|_{i,j,k,l} \in \mathbb{R}^{n_G \times n_{G'} \times n_G \times n_{G'}}$

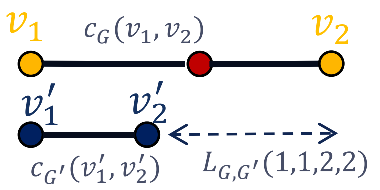
- $M_{G,G'} = \left[\, ||a_i - a'_j||_2 \right]_{1 \le i \le n_G\,;\,1 \le j \le n_{G'}} \in \mathbb{R}^{n_G \times n_{G'}}$

- $FGW_{q,\alpha}(\mu_G, \mu_{G'}) = \min_{\pi \in \Pi} \left\langle \alpha \boxed{M^q_{G,G'}} + (1-\alpha) \boxed{L^q_{G,G'} \otimes \pi}, \quad \pi \right\rangle$

Wasserstein Gromov-Wasserstein

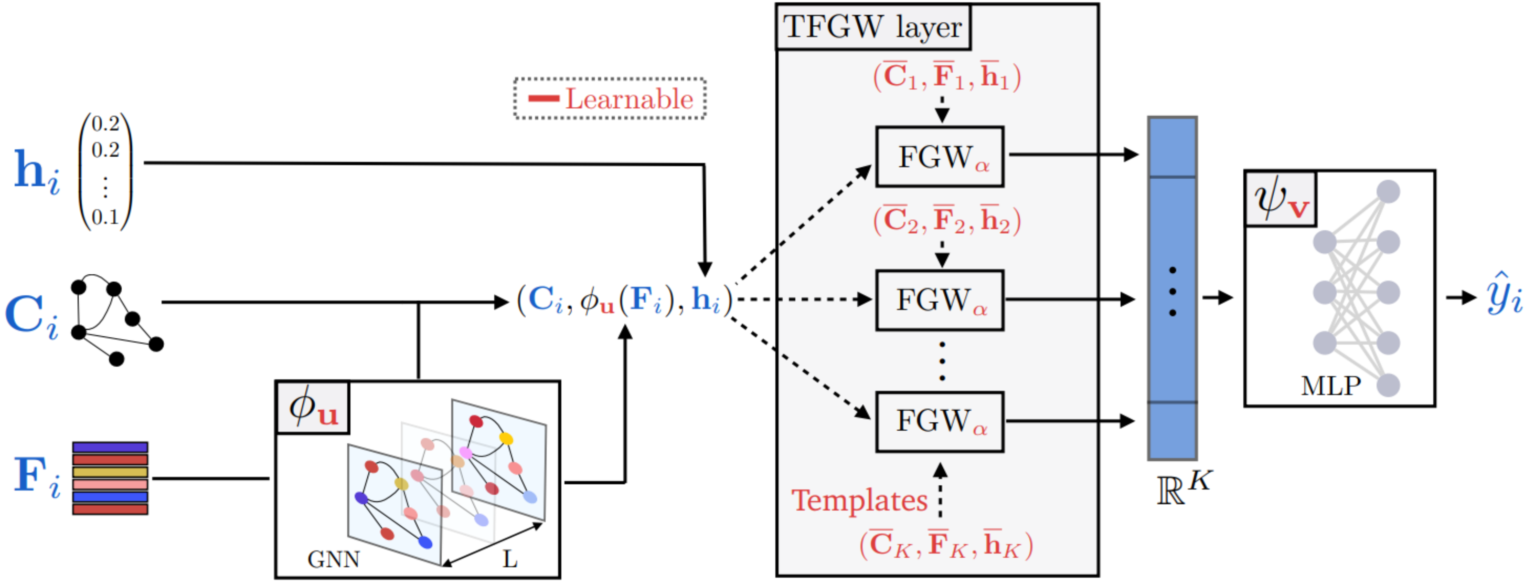
- **Issue**: $\mathrm{k}(G, G') = e^{-\gamma\, FGW_{q,\alpha}(\mu_G, \mu_{G'})}$ is **not positive definite**





Template based GNN with OT

[Vincent-Cuaz et al., 2022]

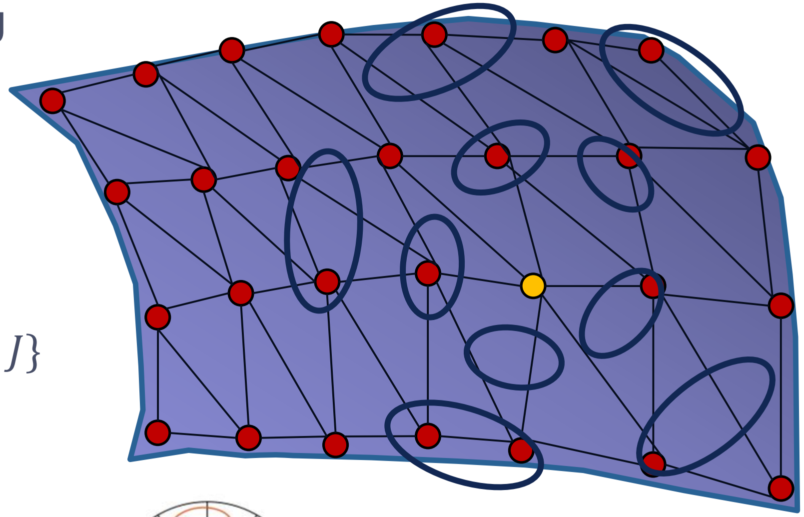


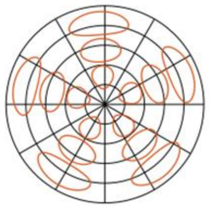
Graph Convolutional Gaussian Processes

[Walker et al., 2019]

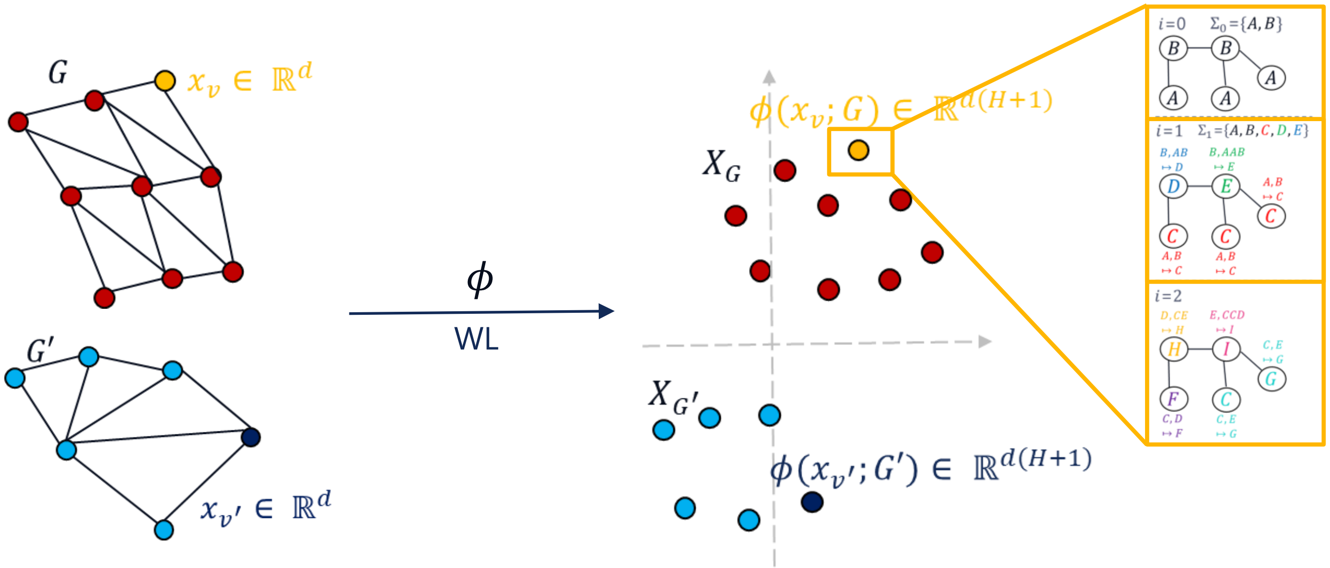
- Graph Convolutional Gaussian Processes
- Local patches around vertices are defined using Spatial-domain charting

- J: number of bins
- Convolution operator on the graph signal $\psi : V \to \mathbb{R}^3$:
- $D_j(v)\,\psi \;=\; \sum_{u \in V} \psi(u)\, u_j\,(u, v) \qquad \forall j \in \{1, \cdots, J\}$
- u_j : geodesic polar weighting function e.g.

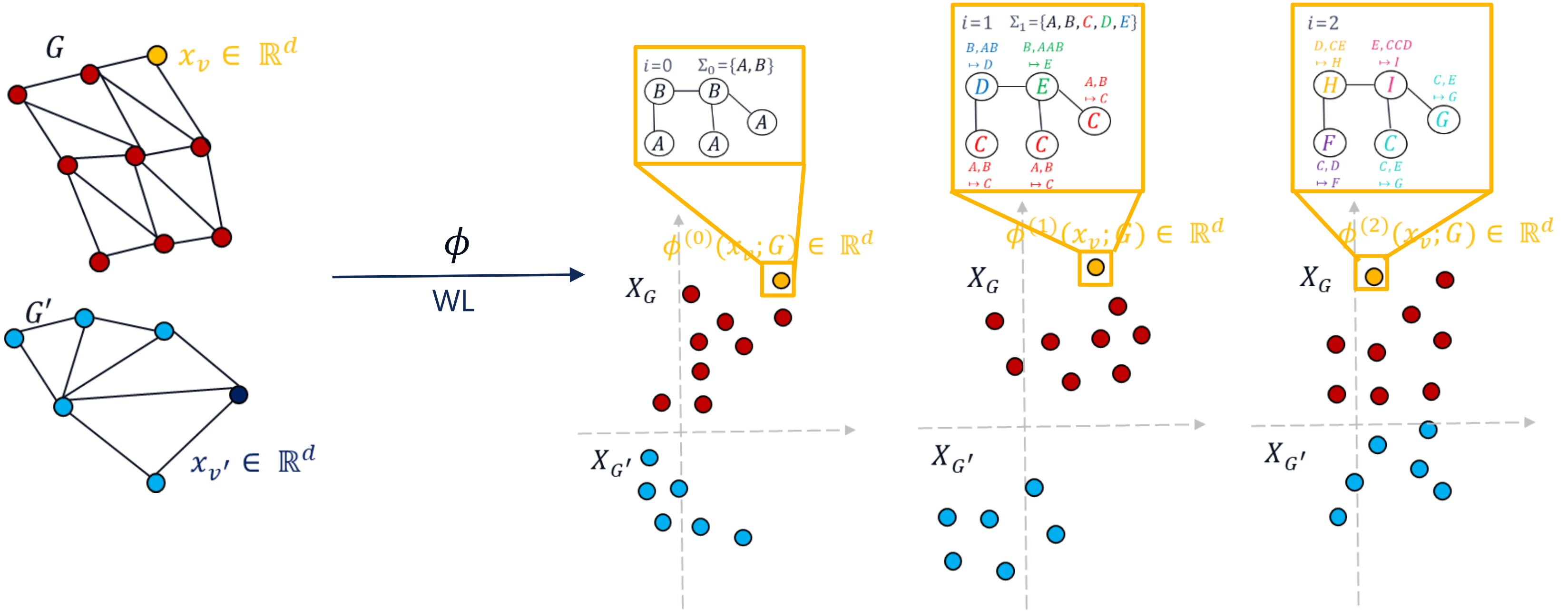




Future work : Anisotropic SWWL?



Future work : Anisotropic SWWL?

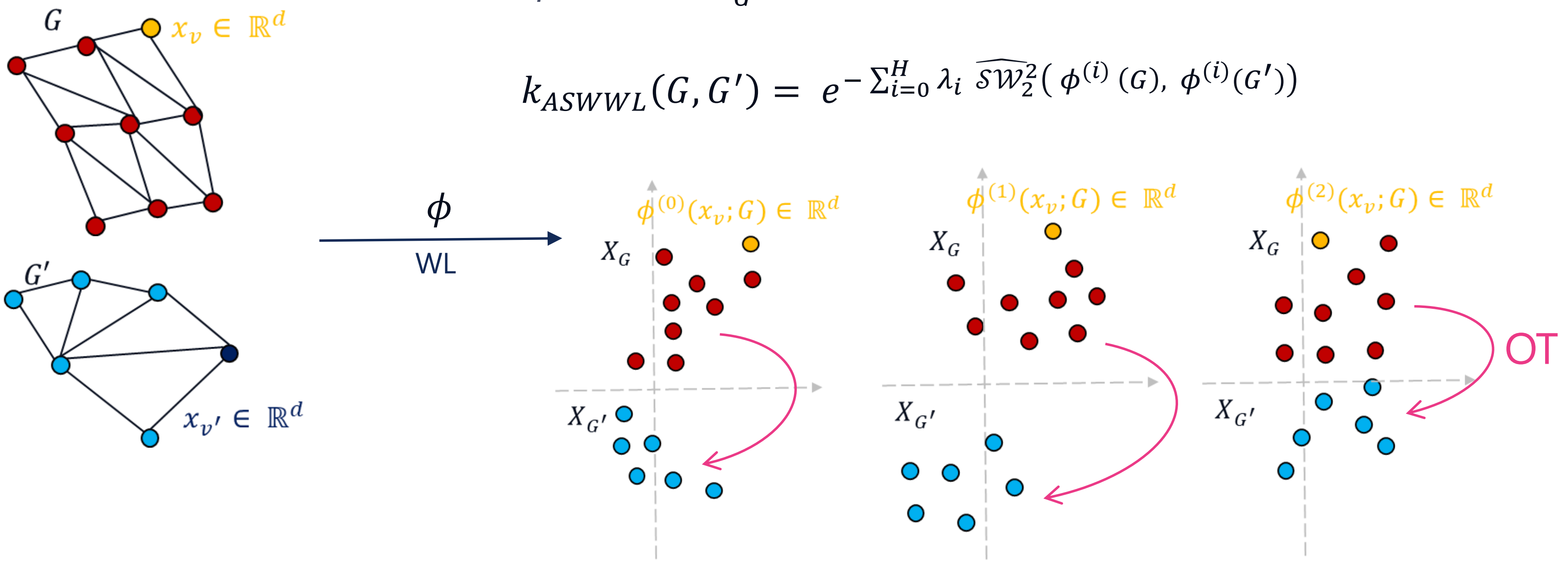


Future work : Anisotropic SWWL?

Anisotropic SWWL:

$$\phi^{(i)} : G \mapsto X_G^{(i)} \in \mathbb{R}^{|V_G| \times d} \text{ (i-th iteration of WL)}$$

$$k_{ASWWL}(G, G') = e^{-\sum_{i=0}^{H} \lambda_i \widehat{SW_2^2}(\phi^{(i)}(G), \phi^{(i)}(G'))}$$



SAFRAN

POWERED BY TRUST

SAFRAN