Graph Coarsening and Graph Neural Networks

Nicolas Keriven

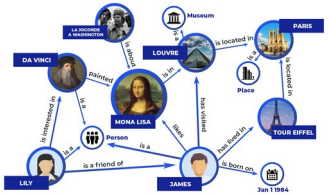
CNRS, IRISA

Joint work with **Antonin Joly** and **Aline Roumy**

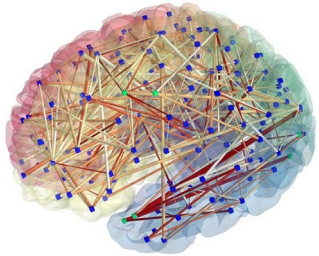
European Research Council
Established by the European Commission

FRANCE 2030
PROGRAMME DE RECHERCHE
INTELLIGENCE ARTIFICIELLE

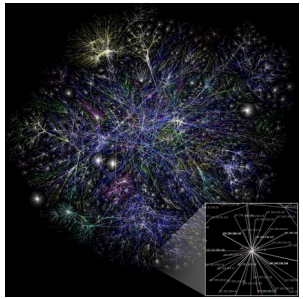
Why care about graphs?



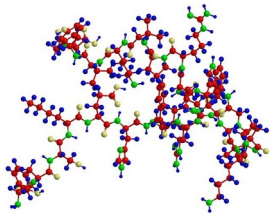
Knowledge graph



Brain connectivity network

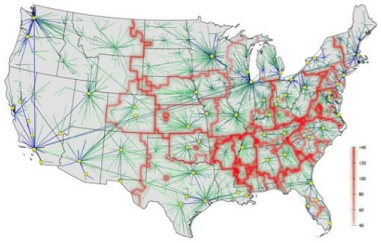


Internet



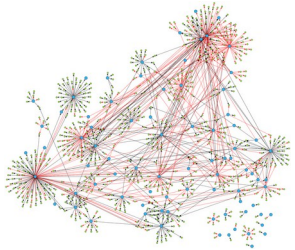
Molecule

Social network

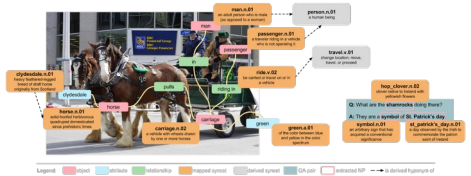


Transportation network

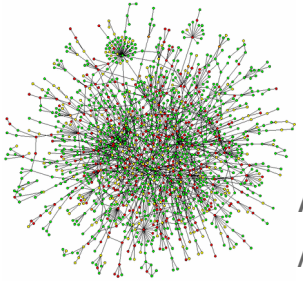
Computer network



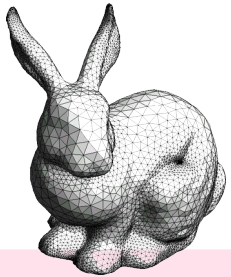
Gene regulatory network



Scene understanding network



Protein interaction network



3D mesh

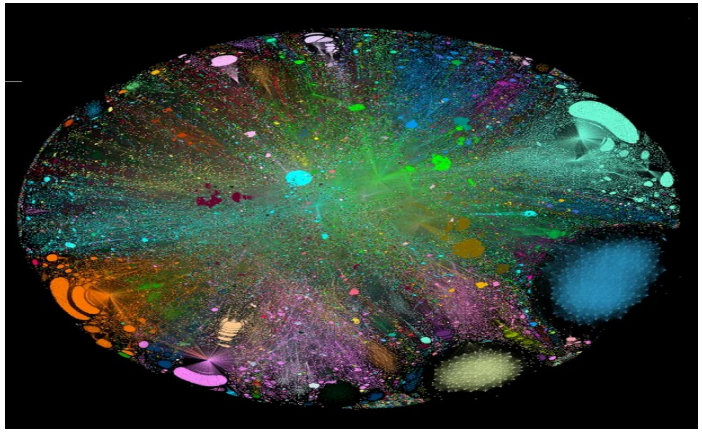
- **Many types of interesting data** can be represented as graphs

  "if all you have is a hammer, everything looks like a nail"

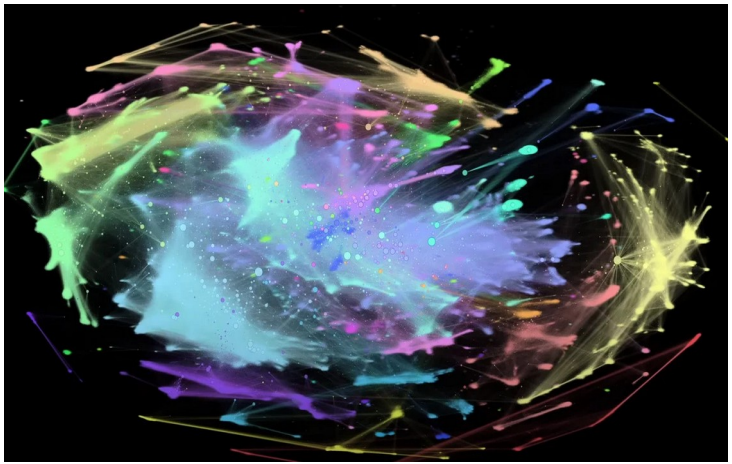
- Modern Graph ML is "part of/referred to as" **Geometric Deep Learning**

Large graphs

- Modern graphs are HUGE
- Any reasonable processing pipeline includes **graph reduction methods**



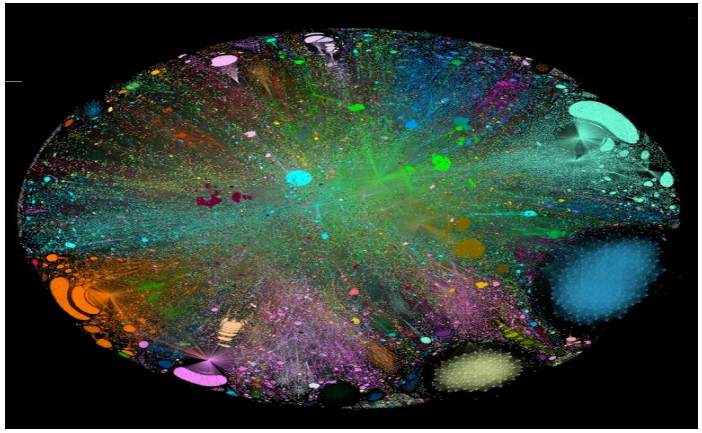
Reddit
200k nodes, 114M edges



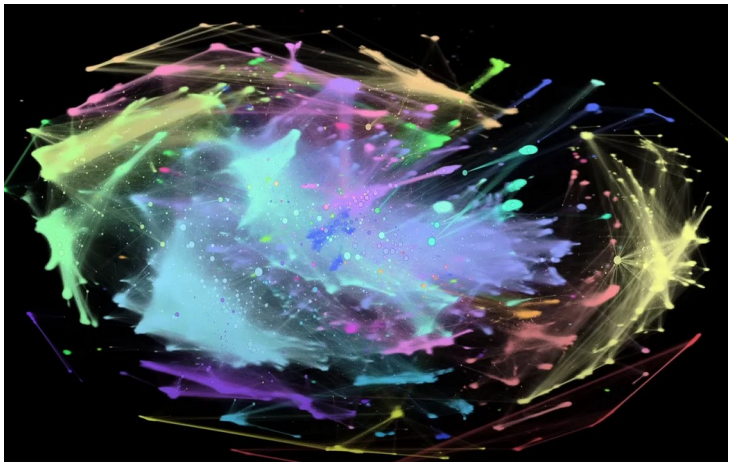
Wikipedia
6M nodes, 200M edges

Large graphs

- Modern graphs are HUGE
- Any reasonable processing pipeline includes **graph reduction methods**
- **Two** main kinds:
  - Graph sampling
  - Graph coarsening



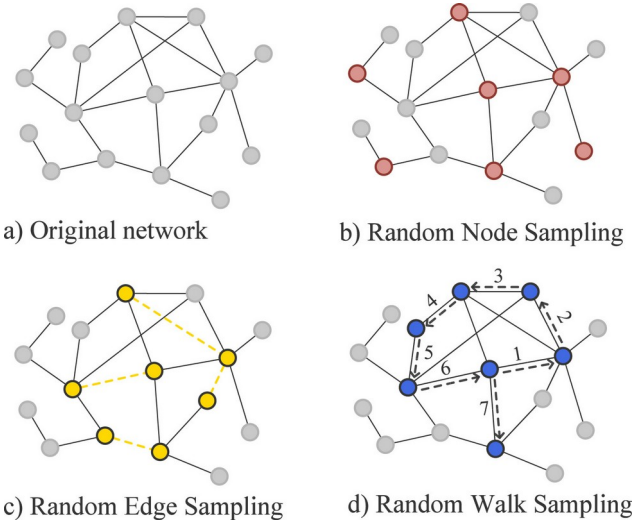
Reddit
200k nodes, 114M edges



Wikipedia
6M nodes, 200M edges

Graph sampling

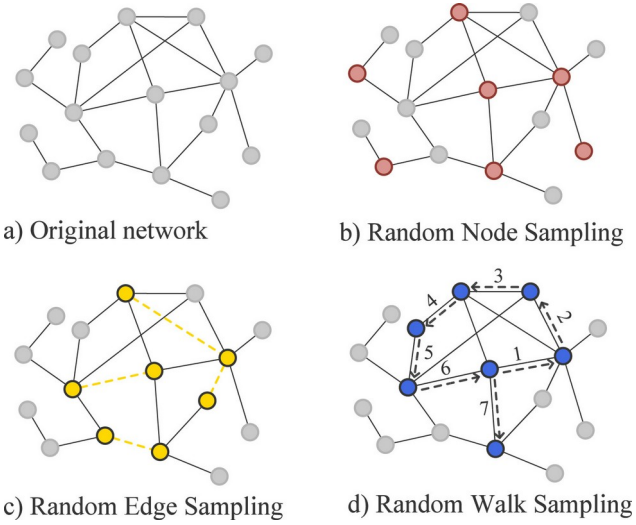
- **Sample** nodes/edges from a large graph

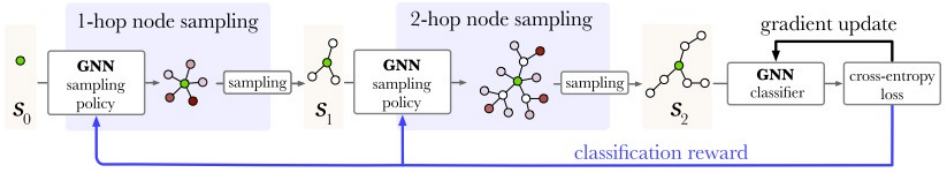


Graph sampling

- **Sample** nodes/edges from a large graph

- **Pros**:
  - Light, fast
  - Can be done at many stages
  - Many different samplers



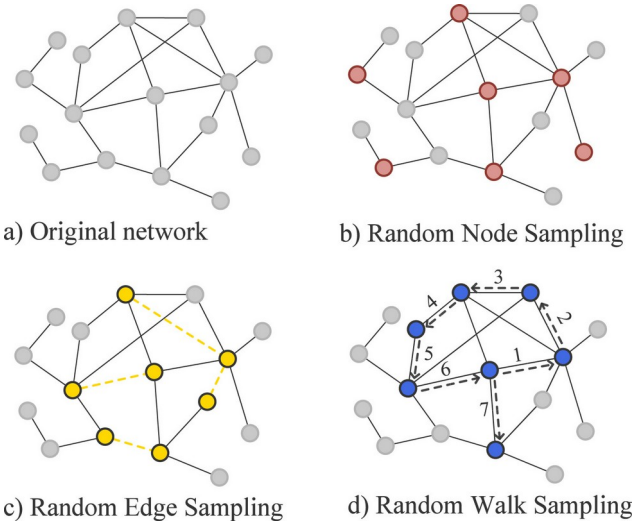


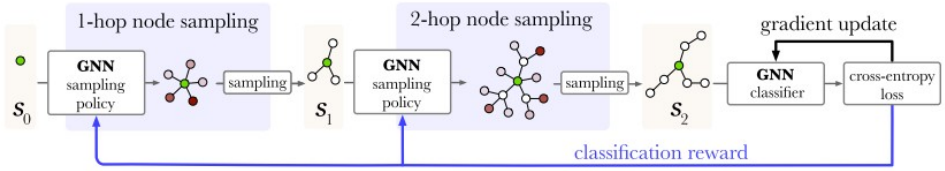
Graph sampling

- **Sample** nodes/edges from a large graph

- **Pros**:
  - Light, fast
  - Can be done at many stages
  - Many different samplers

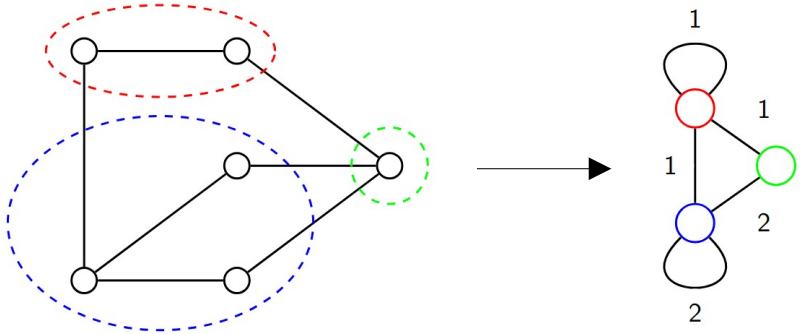
- **Cons**:
  - Extracting a subgraph
  - Difficult to control the loss of information
  - k-hop sampling vs. small-world graphs





Graph coarsening

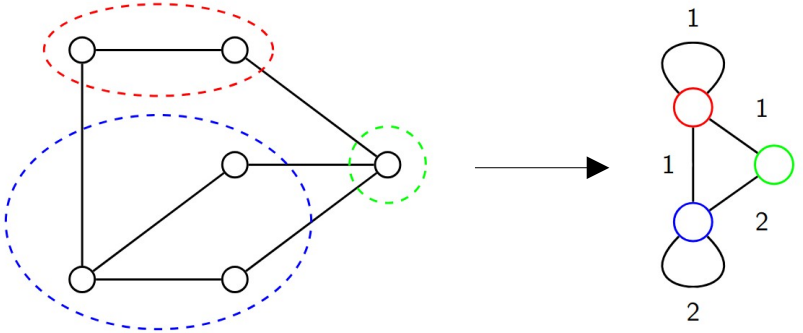
- **Create** a small graph from a large graph

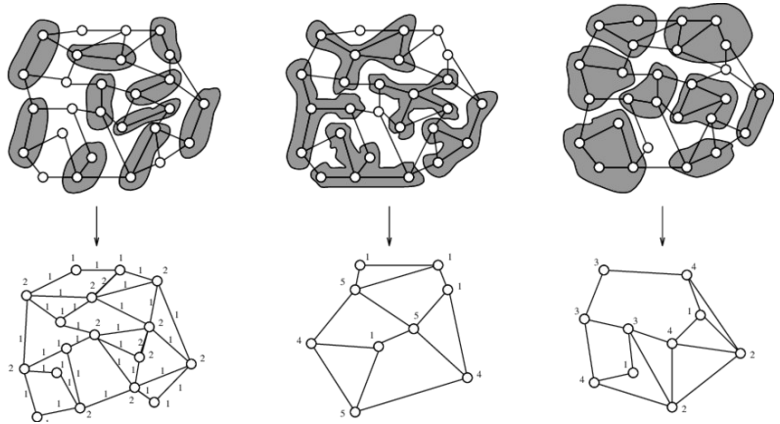


Graph coarsening

- **Create** a small graph from a large graph

- **Pros**:
  - Theoretically more consistent
  - Many algorithms
  - The small graph often already *means* something!

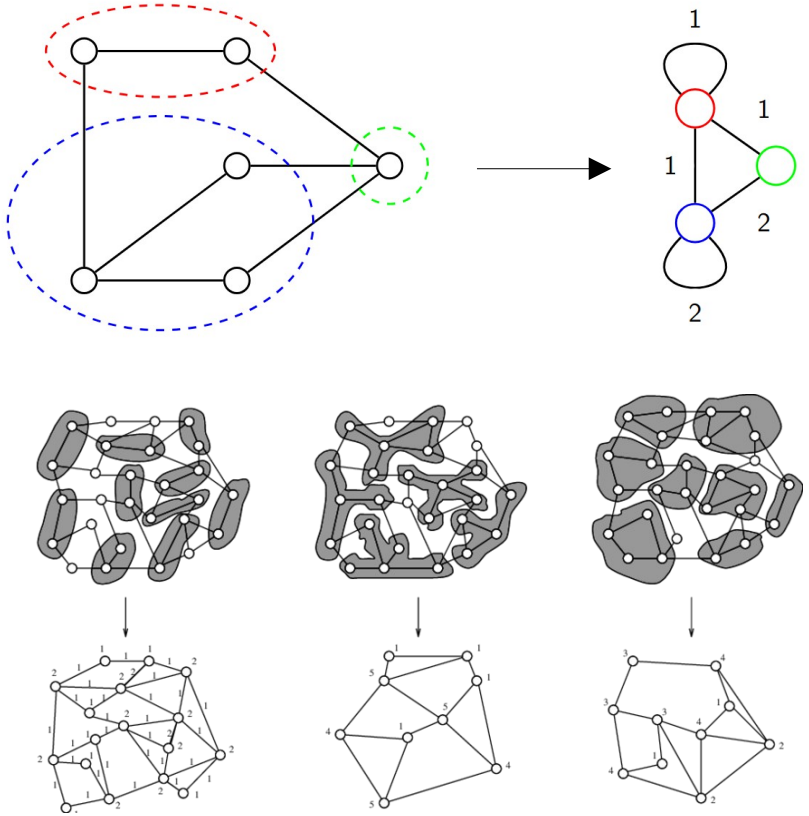




Different ways to coarsen a graph

Graph coarsening

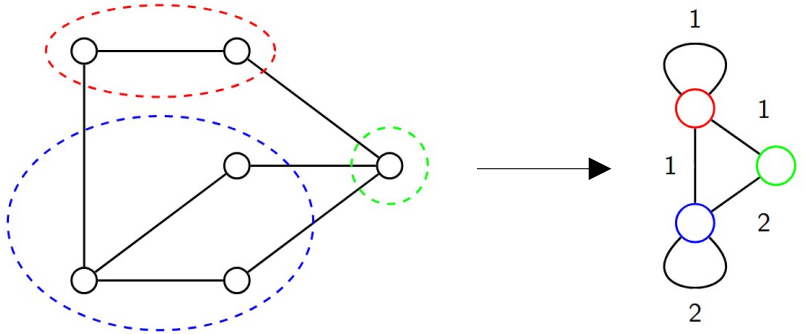
- **Create** a small graph from a large graph
- **Pros**:
  - Theoretically more consistent
  - Many algorithms
  - The small graph often already *means* something!
- **Cons**:
  - Graph "learning" is difficult
  - Computationally costly!

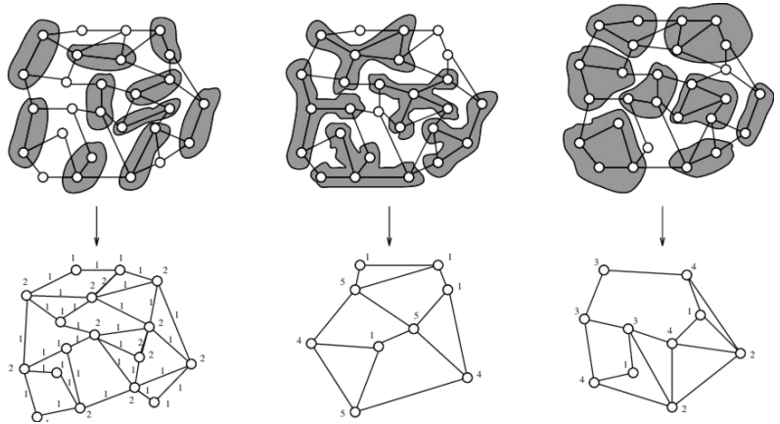


Different ways to coarsen a graph

Graph coarsening

- **Create** a small graph from a large graph
- **Pros**:
  - Theoretically more consistent
  - Many algorithms
  - The small graph often already *means* something!
- **Cons**:
  - Graph "learning" is difficult
  - Computationally cos



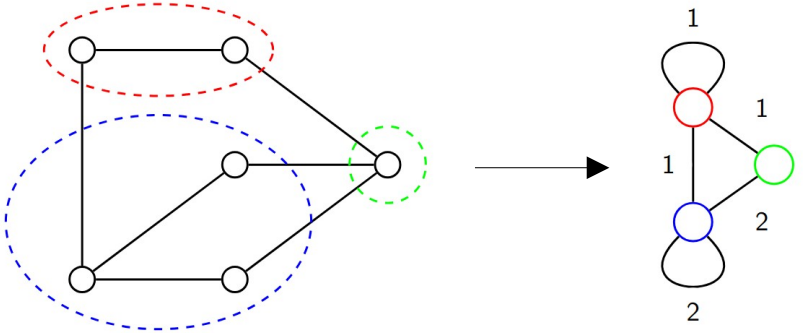


ys to coarsen a graph

This talk: graph coarsening and GNNs

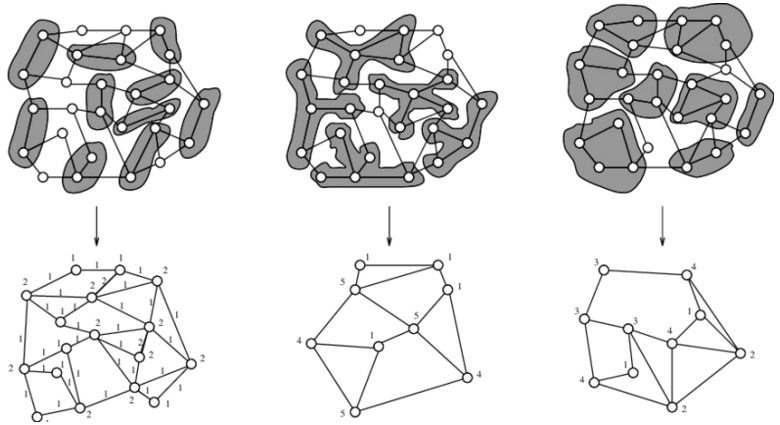
Graph coarsening

This talk: graph coarsening and GNNs



Antonin Joly, Nicolas Keriven, Aline Roumy. **Graph Coarsening with Message-Passing Guarantees** . *NeurIPS 2024*.

Antonin Joly, Nicolas Keriven, Aline Roumy. **Taxonomy of reduction matrices for Graph Coarsening**. *NeurIPS 2025*.



Different ways to coarsen a graph

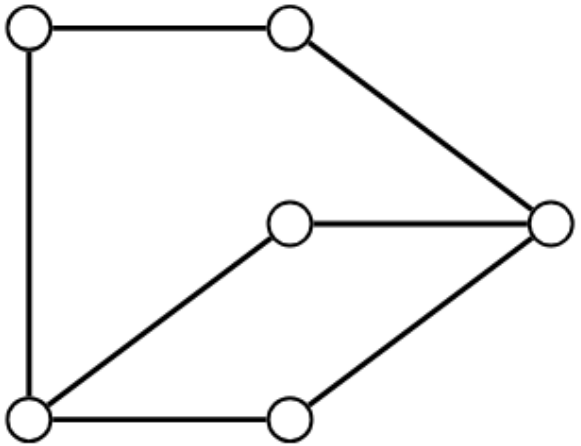
Outline

1. Graph coarsening...
2. ... and Graph Neural Networks
3. Bonus: taxonomy of graph coarsening

Graph

Adjacency matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$



Laplacian

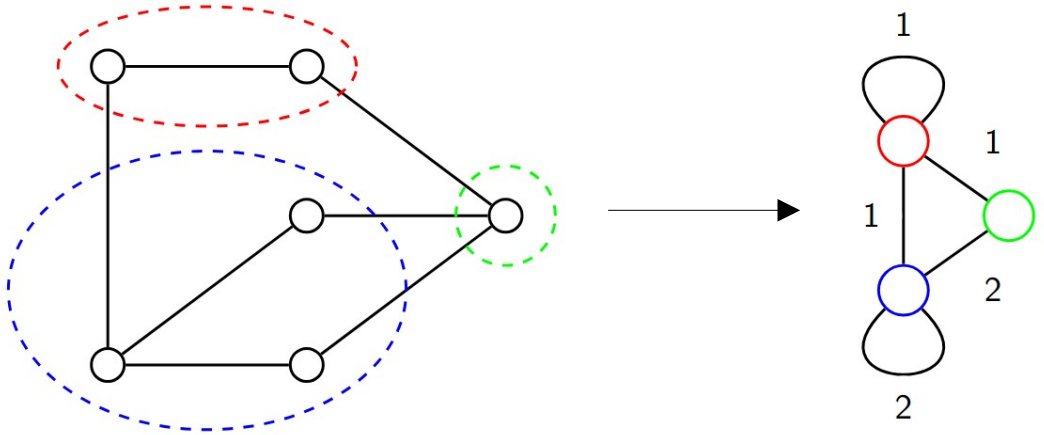
$$L = D - A$$

Diagonal of degrees

Graph coarsening

Graph coarsening:

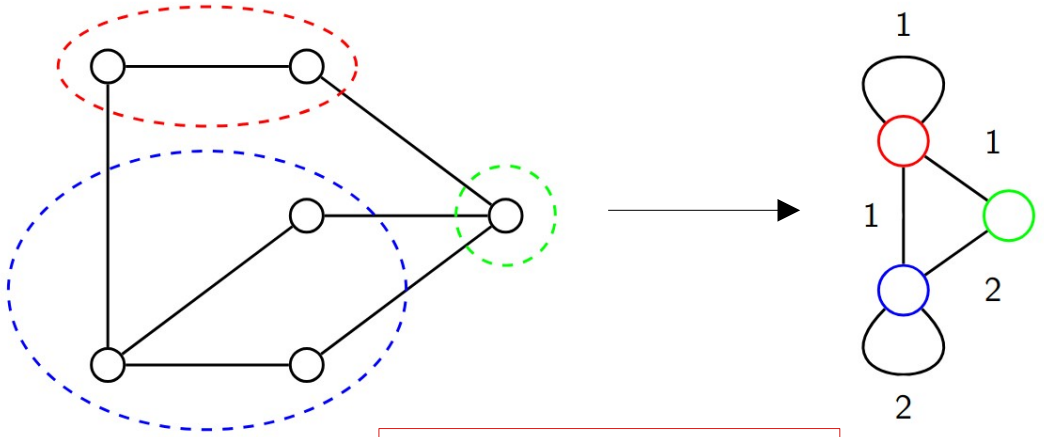
- group nodes into "**supernodes**"
- Create weighted edges by counting original ones



Graph coarsening

Graph coarsening:

- group nodes into "**supernodes**"
- Create weighted edges by counting original ones



Coarsened Adjacency matrix

$$A_c = Q^\top A Q$$

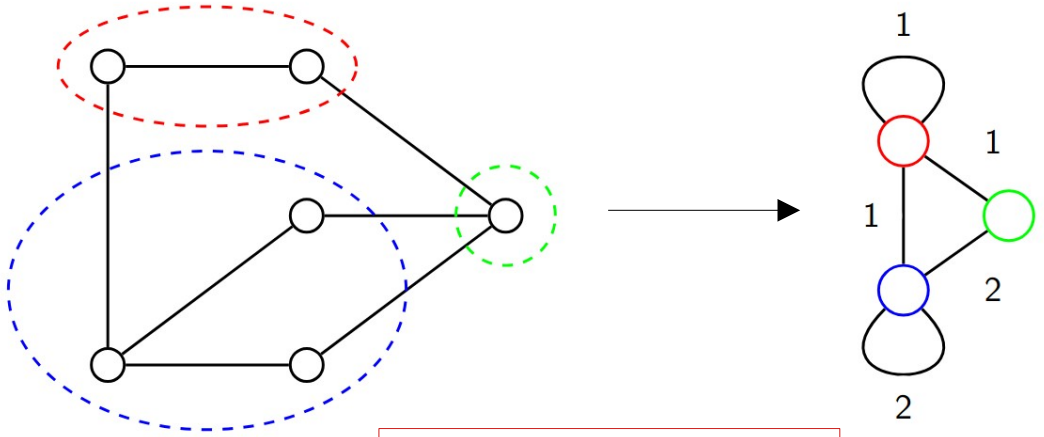
$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Membership matrix

Graph coarsening

Graph coarsening:

- group nodes into "**supernodes**"
- Create weighted edges by counting original ones



Coarsened Adjacency matrix

$$A_c = Q^\top A Q$$

Coarsened Laplacian

$$L_c = D_c - A_c = Q^\top L Q$$

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

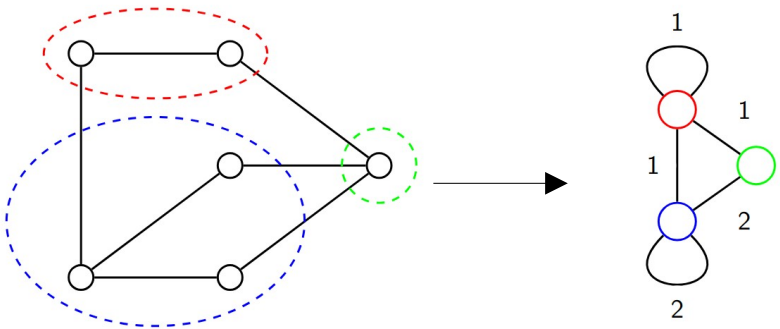
Membership matrix

Graph coarsening consistency

Lemma. (Consistency of Laplacian [1])
Let Q be a well-partitioned matrix. The two following properties are equivalent:

$$Q \text{ is proportional to a binary matrix.}$$
$$\Updownarrow$$
$$\forall A, \quad L(A_c) = Q^\top L Q$$

[1] Andreas Loukas, *Graph Reduction with Spectral and Cut Guarantees*, JMLR 2019.

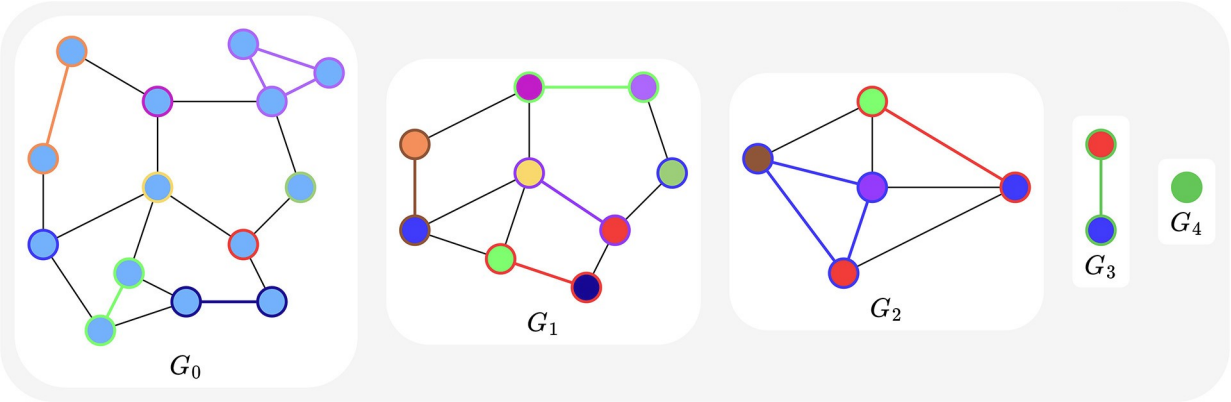


$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Membership matrix

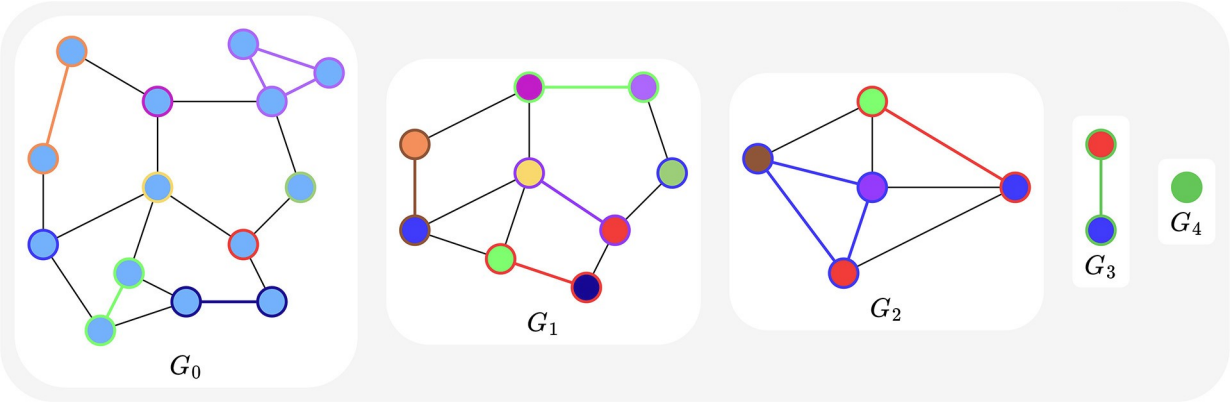
How to coarsen a graph?

- Many different methods
- Often ***iteratively***
  - Select "contracting set"
    - edges, neighborhood...
  - Merge them
  - Repeat



How to coarsen a graph?

- Many different methods
- Often ***iteratively***
  - Select "contracting set"
    - edges, neighborhood...
  - Merge them
  - Repeat



- Generally done to (approximately) optimize some ***cost,*** in a **greedy** manner

Restricted Spectral Approximation

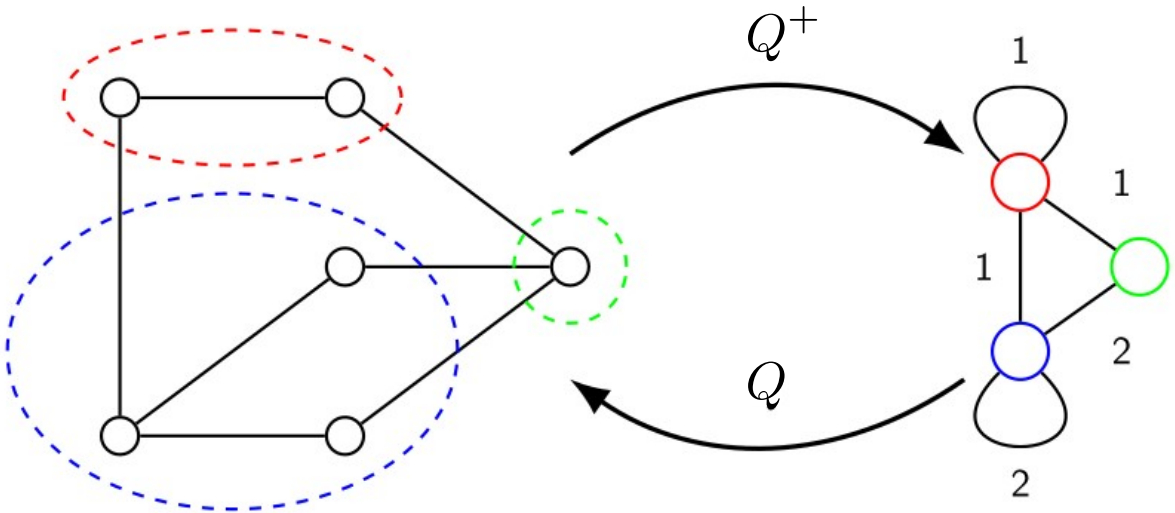
Intuitively, possible to preserve the **low-frequencies** of the graph

- Slow varying ***signals*** $x \in \mathbb{R}^n$ on the big graph can be well-approximated on the small graph

Restricted Spectral Approximation

Intuitively, possible to preserve the **low-frequencies** of the graph

- Slow varying ***signals*** $x \in \mathbb{R}^n$ on the big graph can be well-approximated on the small graph



Reduction

$$x_c = Q^+ x$$

Average within super-nodes

Lifting

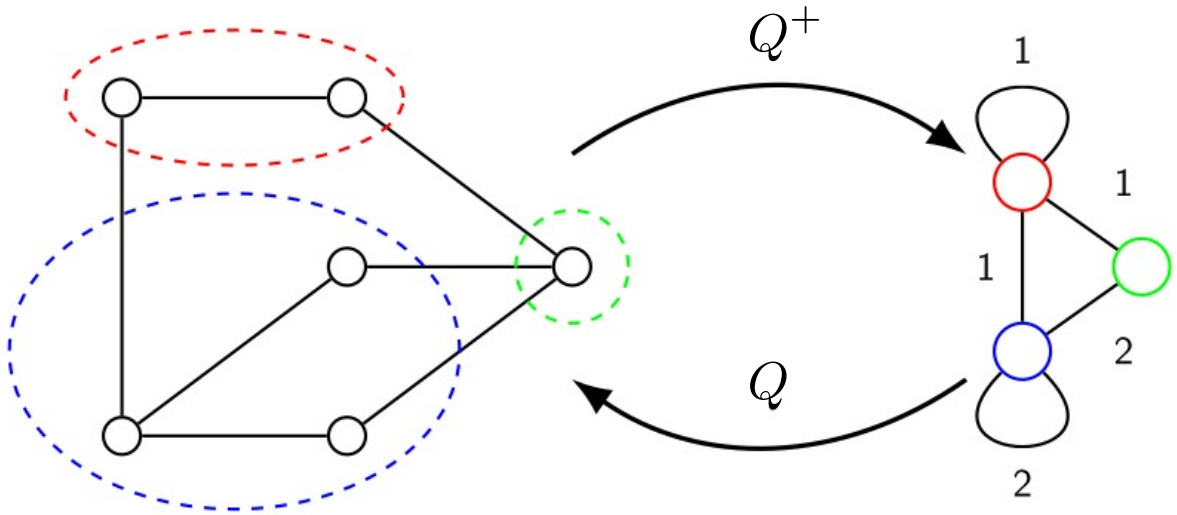
$$\tilde{x} = Q x_c$$

Expand constant signal within super-nodes

Restricted Spectral Approximation

Intuitively, possible to preserve the **low-frequencies** of the graph

- Slow varying ***signals*** $x \in \mathbb{R}^n$ on the big graph can be well-approximated on the small graph



Reduction

$$x_c = Q^+ x$$

Average within super-nodes

Lifting

$$\tilde{x} = Q x_c$$

Expand constant signal within super-nodes

Does ***reduction-lifting*** degrades too much the signal?

Restricted Spectral Approximation

Speed of variation (aka Dirichlet energy): $$x^\top L x = \sum_{i \sim j} (x_i - x_j)^2 = \|x\|_L^2$$

Restricted Spectral Approximation

Speed of variation (aka Dirichlet energy): $x^\top Lx = \sum_{i\sim j}(x_i - x_j)^2 = \|x\|_L^2$

RSA: Bound difference between *signal* and **projected-reconstructed version** $\begin{cases} x_c &= Q^+ x \\ \tilde{x} &= Q x_c \end{cases}$

Restricted Spectral Approximation

Speed of variation (aka Dirichlet energy): $x^\top Lx = \sum_{i\sim j}(x_i - x_j)^2 = \|x\|_L^2$

RSA: Bound difference between *signal* and **projected-reconstructed version** $\begin{cases} x_c &= Q^+ x \\ \tilde{x} &= Q x_c \end{cases}$

$$\epsilon_Q = \max_{x\in\mathcal{R}, \|x\|_L=1} \|x - \tilde{x}\|_L$$

some *target* subspace

Consistency: $\|\tilde{x}\|_L = \|x_c\|_{L_c}$

Coarsening algorithm

Goal: $\min_{Q} \epsilon_Q$

Coarsening algorithm

Goal: $\min_{Q} \epsilon_Q$

- NP-hard in general

Coarsening algorithm

Goal: $\min_{Q} \epsilon_Q$

- NP-hard in general
- Approximative algo, often **greedy**
  - Might still be quite costly!

Loukas. *Graph reduction with spectral and cut guarantees* (2019)

Algorithm 2 Single-level coarsening by local variation

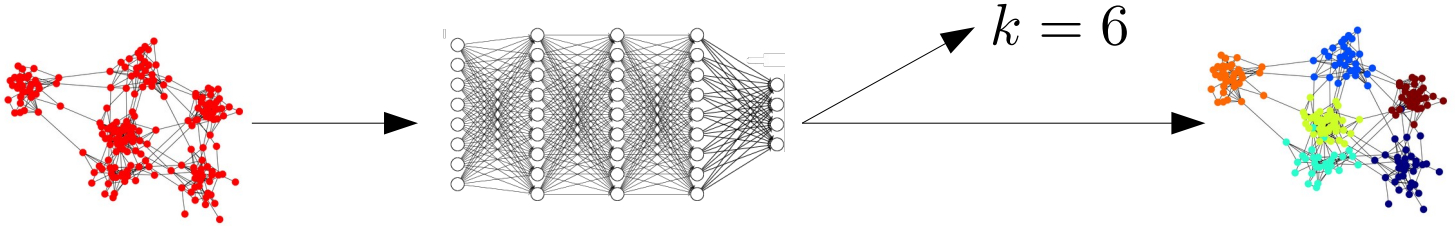
input: Combinatorial Laplacian $L_{\ell-1}$, threshold σ', and target size n.
Form the family of candidate sets $\mathcal{F}_\ell = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \ldots\}$ (algorithm-specific step).
$N_\ell \leftarrow N_{\ell-1}$, marked $\leftarrow \varnothing$, $\sigma_\ell^2 \leftarrow 0$.
Sort $\mathcal{F}_\ell$ in terms of increasing $\text{cost}_\ell(\mathcal{C})$.
while $|\mathcal{F}_\ell| > 0$ and $N_\ell > n$ and $\sigma_\ell \leq \sigma'$ **do**
    Pop the candidate set $\mathcal{C}$ of minimal cost s from $\mathcal{F}_\ell$.
    if all vertices of $\mathcal{C}$ are not marked and $\sigma' \geq \sqrt{\sigma_\ell^2 + (|\mathcal{C}| - 1)s}$ **then**
        marked $\leftarrow$ marked $\cup\, \mathcal{C}$, $\mathcal{P}_\ell \leftarrow \mathcal{P}_\ell \cup \mathcal{C}$, $N_\ell \leftarrow N_\ell - |\mathcal{C}| + 1$, $\sigma_\ell^2 \leftarrow \sigma_\ell^2 + (|\mathcal{C}| - 1)s$
    else
        $\mathcal{C}' \leftarrow \mathcal{C} \setminus$ marked
        if $|\mathcal{C}'| > 1$ **then**
            Compute $\text{cost}_\ell(\mathcal{C}')$ and insert $\mathcal{C}'$ into $\mathcal{F}_\ell$ while keeping the latter sorted.
Form the $N_\ell \times N_{\ell-1}$ coarsening matrix P_ℓ based on $\mathcal{P}_\ell$.
return $L_\ell \leftarrow P_\ell^{\mp} L_{\ell-1} P_\ell^{+}$ and σ_ℓ

Proposition 17 *Matrices L_c and L are $(\mathbf{R}, \epsilon)$-similar with $\epsilon \leq \prod_{\ell=1}^{c}(1 + \sigma_\ell) - 1$.*

Outline

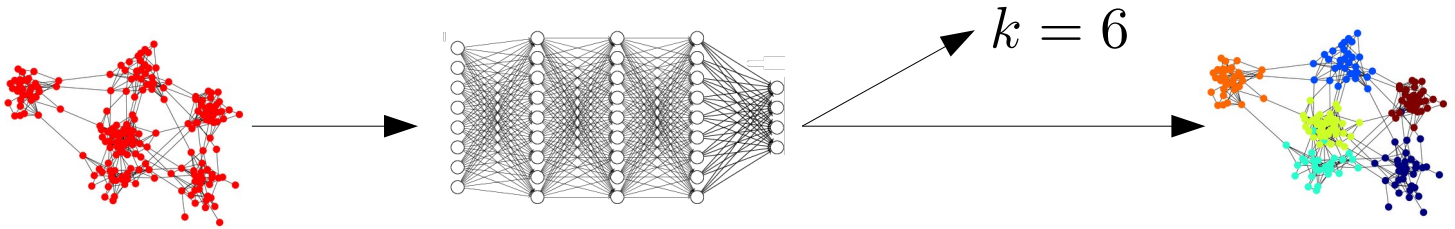
1. Graph coarsening...
2. ... and Graph Neural Networks
3. Bonus: taxonomy of graph coarsening

ML on graphs: Graph Neural Networks



Graph Neural Networks (GNN) are "deep architectures" to do ML on graphs.

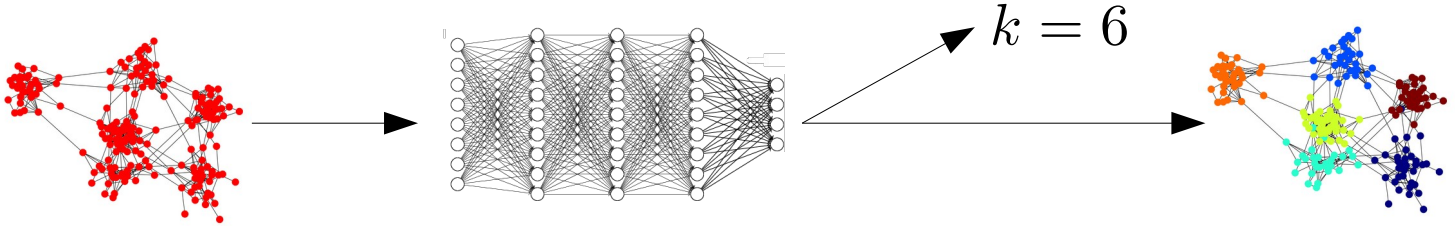
ML on graphs: Graph Neural Networks



Graph Neural Networks (GNN) are "deep architectures" to do ML on graphs.

- Updates ***node representations*** at each layer, starting with input **node features**
  - No node features? Handcrafted stuff, but not ideal...

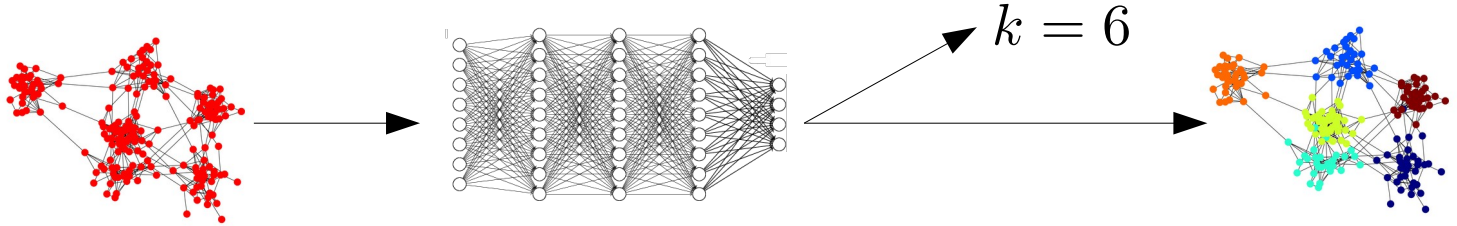
ML on graphs: Graph Neural Networks



Graph Neural Networks (GNN) are "deep architectures" to do ML on graphs.

- Updates ***node representations*** at each layer, starting with input **node features**
  - No node features? Handcrafted stuff, but not ideal...
- Alternates linear operations/non-linear activation functions

ML on graphs: Graph Neural Networks

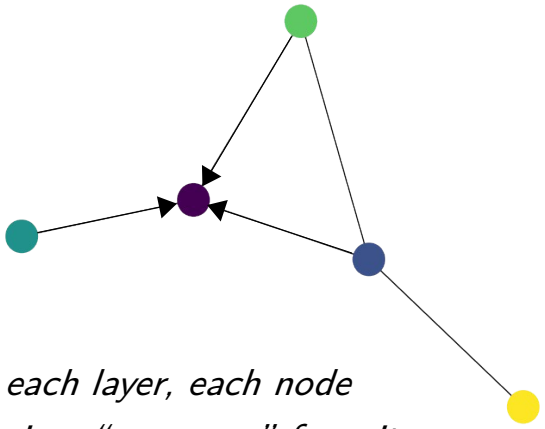


Graph Neural Networks (GNN) are "deep architectures" to do ML on graphs.

- Updates ***node representations*** at each layer, starting with input **node features**
  - No node features? Handcrafted stuff, but not ideal...
- Alternates linear operations/non-linear activation functions
- Extremely ***flexible***: can perform node/graph/link prediction with minimal adaptation
- State-of-the-art, but still quite many problems

The message-passing paradigm

Modern GNNs are often built around a message-passing interpretation.

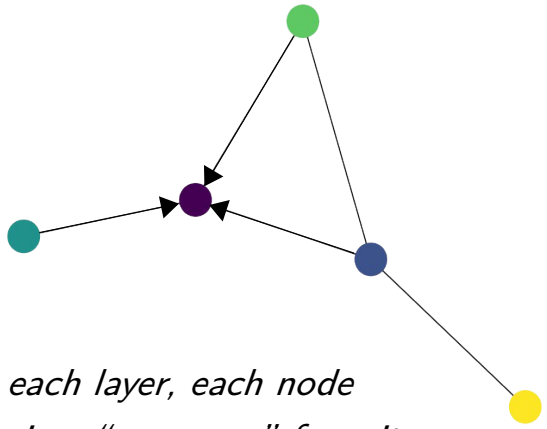


At each layer, each node receives "messages" from its neighbors.

Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)
Kipf et al. *Semi-Supervised Learning with Graph Convolutional Networks* (2017)

The message-passing paradigm

Modern GNNs are often built around a message-passing interpretation.



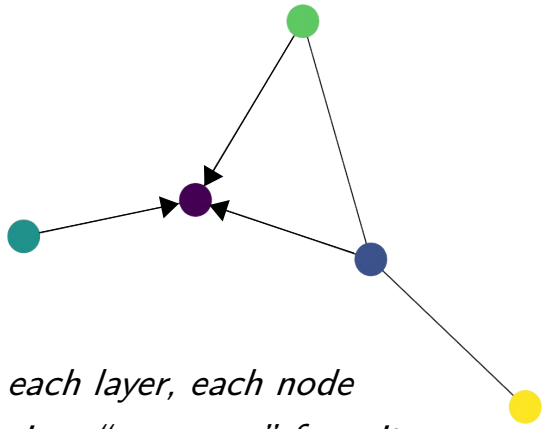
At each layer, each node receives "messages" from its neighbors.

Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)
Kipf et al. *Semi-Supervised Learning with Graph Convolutional Networks* (2017)

- $x_i^{(0)}, 1 \leq i \leq n$ *Initial node features*
- $x_i^{(\ell)} = \text{AGGREGATE}\left(x_i^{(\ell-1)}, \{x_j^{(\ell-1)}, a_{ij}\}\right)$
- $\text{output} = \begin{cases} \{x_i^{(L)}\} & \text{(node prediction)} \\ \text{POOL}(\{x_i^{(L)}\}_i) & \text{(graph prediction)} \end{cases}$

The message-passing paradigm

Modern GNNs are often built around a message-passing interpretation.



At each layer, each node receives "messages" from its neighbors.

Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)
Kipf et al. *Semi-Supervised Learning with Graph Convolutional Networks* (2017)

- $x_i^{(0)}, 1 \leq i \leq n$ *Initial node features*
- $x_i^{(\ell)} = \text{AGGREGATE}\left(x_i^{(\ell-1)}, \{x_j^{(\ell-1)}, a_{ij}\}\right)$
- $\text{output} = \begin{cases} \{x_i^{(L)}\} & \text{(node prediction)} \\ \text{POOL}(\{x_i^{(L)}\}_i) & \text{(graph prediction)} \end{cases}$

- Messages are treated as a **set**: no node ordering!
- Resulting GNNs are **permutation equivariant** (node) or **invariant** (graph), and can be applied to ***any graph*** once trained

Propagation matrix

Message-passing often materialized through a ***propagation matrix*** $S \in \mathbb{R}^{n \times n}$, such that

$$X^{(\ell)} = \sigma\left(SX^{(\ell-1)}W^{(\ell)} + 1_n b_\ell^\top\right)$$

Propagation matrix

Message-passing often materialized through a ***propagation matrix*** $S \in \mathbb{R}^{n \times n}$, such that

$$X^{(\ell)} = \sigma\left(SX^{(\ell-1)}W^{(\ell)} + 1_n b_\ell^\top\right)$$

Node representations

$X^{(\ell)} \in \mathbb{R}^{n \times d_\ell}$

Propagation matrix

Message-passing often materialized through a ***propagation matrix*** $S \in \mathbb{R}^{n \times n}$, such that

$$X^{(\ell)} = \sigma\left(SX^{(\ell-1)}W^{(\ell)} + 1_n b_\ell^\top\right)$$

Node representations $X^{(\ell)} \in \mathbb{R}^{n \times d_\ell}$

Non-linear activation function

Propagation matrix

Message-passing often materialized through a ***propagation matrix*** $S \in \mathbb{R}^{n \times n}$, such that

$$X^{(\ell)} = \sigma\left(SX^{(\ell-1)}W^{(\ell)} + 1_n b_\ell^\top\right)$$

Node representations
$X^{(\ell)} \in \mathbb{R}^{n \times d_\ell}$

Non-linear activation function

Learnable weights and bias
$W^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$
$b_\ell \in \mathbb{R}^{d_\ell}$

Propagation matrix

Message-passing often materialized through a ***propagation matrix*** $S \in \mathbb{R}^{n \times n}$, such that

$$X^{(\ell)} = \sigma\left(S X^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

Node representations
$X^{(\ell)} \in \mathbb{R}^{n \times d_\ell}$

Non-linear activation function

Learnable weights and bias
$W^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$
$b_\ell \in \mathbb{R}^{d_\ell}$

Example:

$$S = A \rightarrow x_i^{(\ell)} = \sigma\left({W^{(\ell)}}^\top \sum_{j \sim i} x_j^{(\ell-1)} + b_\ell\right) \qquad \textit{(sum aggregation)}$$

Propagation matrix

Message-passing often materialized through a ***propagation matrix*** $S \in \mathbb{R}^{n \times n}$, such that

$$X^{(\ell)} = \sigma\left(S X^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

Node representations $X^{(\ell)} \in \mathbb{R}^{n \times d_\ell}$

Non-linear activation function

Learnable weights and bias $W^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$, $b_\ell \in \mathbb{R}^{d_\ell}$

Example:

$$S = A \rightarrow x_i^{(\ell)} = \sigma\left(W^{(\ell)^\top} \sum_{j \sim i} x_j^{(\ell-1)} + b_\ell\right) \quad \textit{(sum aggregation)}$$

$$S = D^{-1}A \rightarrow x_i^{(\ell)} = \sigma\left(W^{(\ell)^\top} \frac{1}{d_i} \sum_{j \sim i} x_j^{(\ell-1)} + b_\ell\right) \quad \textit{(mean aggregation)}$$

Propagation matrix

Message-passing often materialized through a ***propagation matrix*** $S \in \mathbb{R}^{n \times n}$, such that

$$X^{(\ell)} = \sigma\left(S X^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

Node representations $X^{(\ell)} \in \mathbb{R}^{n \times d_\ell}$

Non-linear activation function

Learnable weights and bias $W^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$, $b_\ell \in \mathbb{R}^{d_\ell}$

Example:

$$S = A \to x_i^{(\ell)} = \sigma\left(W^{(\ell)\top} \sum_{j \sim i} x_j^{(\ell-1)} + b_\ell\right) \quad \textit{(sum aggregation)}$$

$$S = D^{-1}A \to x_i^{(\ell)} = \sigma\left(W^{(\ell)\top} \frac{1}{d_i} \sum_{j \sim i} x_j^{(\ell-1)} + b_\ell\right) \quad \textit{(mean aggregation)}$$

$$S = D^{-1/2}AD^{-1/2} \to x_i^{(\ell)} = \sigma\left(W^{(\ell)\top} \frac{1}{\sqrt{d_i}} \sum_{j \sim i} \frac{x_j^{(\ell-1)}}{\sqrt{d_j}} + b_\ell\right)$$

GNN and graph reduction

One application of GNN: $O(L(n + |E|))$ ***Training**: times number of iterations!*

GNN and graph reduction

One application of GNN: $O(L(n + |E|))$ ***Training**: times number of iterations!*

Graph reduction (sampling, coarsening):

- As **preprocessing**
  - Huang et al. *Scaling up Graph Neural Networks Via Graph Coarsening*, 2021

This paper

GNN and graph reduction

One application of GNN: $O(L(n + |E|))$ ***Training**: times number of iterations!*

Graph reduction (sampling, coarsening):

- As **preprocessing**
  - Huang et al. *Scaling up Graph Neural Networks Via Graph Coarsening*, 2021

This paper

- During training (**batching**...)
  - Gasteiger et al. *Influence-Based Mini-Batching for Graph Neural Networks*. 2022

GNN and graph reduction

One application of GNN: $O(L(n + |E|))$ ***Training**: times number of iterations!*

Graph reduction (sampling, coarsening):

- As **preprocessing**
  - Huang et al. *Scaling up Graph Neural Networks Via Graph Coarsening*, 2021

This paper

- During training (**batching**...)
  - Gasteiger et al. *Influence-Based Mini-Batching for Graph Neural Networks*. 2022
- **Within** the GNN itself -> may be differentiable
  - Ying et al. *Hierarchical graph representation learning with differentiable pooling*. 2018

GNN and graph reduction

One application of GNN: $O(L(n + |E|))$ ***Training**: times number of iterations!*

Graph reduction (sampling, coarsening):

- As **preprocessing**
  - Huang et al. *Scaling up Graph Neural Networks Via Graph Coarsening*, 2021

This paper

- During training (**batching**...)
  - Gasteiger et al. *Influence-Based Mini-Batching for Graph Neural Networks*. 2022
- **Within** the GNN itself -> may be differentiable
  - Ying et al. *Hierarchical graph representation learning with differentiable pooling*. 2018
- GNN can also be *trained* to coarsen graphs
  - Cai et al. *Graph coarsening with neural networks.* 2021

GNN and graph reduction

Q: Does training on a coarsened graph **approximate training** on the original graph?

GNN and graph reduction

Q: Does training on a coarsened graph **approximate training** on the original graph?

Original

$$X^{(0)} \in \mathbb{R}^{n \times d_0}$$

$$X^{(\ell)} = \sigma\left(S X^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

Coarsened

$$X_c^{(0)} = Q^+ X^{(0)} \in \mathbb{R}^{n_c \times d_0}$$

$$X_c^{(\ell)} = \sigma\left(S_c X_c^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

"coarsened propagation"?

GNN and graph reduction

Q: Does training on a coarsened graph **approximate training** on the original graph?

Original

$$X^{(0)} \in \mathbb{R}^{n \times d_0}$$
$$X^{(\ell)} = \sigma\left(S X^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

Coarsened

$$X_c^{(0)} = Q^+ X^{(0)} \in \mathbb{R}^{n_c \times d_0}$$
$$X_c^{(\ell)} = \sigma\left(S_c X_c^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

"coarsened propagation"?

Approximately the same output?

GNN and graph reduction

Q: Does training on a coarsened graph **approximate training** on the original graph?

Original

$$X^{(0)} \in \mathbb{R}^{n \times d_0}$$
$$X^{(\ell)} = \sigma\left(S X^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

Coarsened

$$X_c^{(0)} = Q^+ X^{(0)} \in \mathbb{R}^{n_c \times d_0}$$
$$X_c^{(\ell)} = \sigma\left(S_c X_c^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

"coarsened propagation"?

Approximately the same output?

- Assuming a good RSA? (good quality coarsening)
- Assuming that all signals are approximately "low-frequencies"?

GNN and graph reduction

Q: Does training on a coarsened graph **approximate training** on the original graph?

Original

$$X^{(0)} \in \mathbb{R}^{n \times d_0}$$
$$X^{(\ell)} = \sigma\left(S X^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

Coarsened

$$X_c^{(0)} = Q^+ X^{(0)} \in \mathbb{R}^{n_c \times d_0}$$
$$X_c^{(\ell)} = \sigma\left(S_c X_c^{(\ell-1)} W^{(\ell)} + 1_n b_\ell^\top\right)$$

"coarsened propagation"?

Approximately the same output?

- Assuming a good RSA? (good quality coarsening)
- Assuming that all signals are approximately "low-frequencies"?

Boils down to: is propagating with S approximately the same as propagating with S_c ?

Choice of coarsened propagation

Natural/previous choice: if $S = f(A)$, take $S_c = f(A_c)$

Choice of coarsened propagation

Natural/previous choice: if $S = f(A)$, take $S_c = f(A_c)$

- That is, $S = A \to S_c = A_c$, $S = L \to S_c = L_c$, etc.
- Basically ignores that the small graph ***comes from a coarsening process***

Choice of coarsened propagation

Natural/previous choice: if $S = f(A)$, take $S_c = f(A_c)$

- That is, $S = A \to S_c = A_c$, $S = L \to S_c = L_c$, etc.
- Basically ignores that the small graph ***comes from a coarsening process***

Does not work! Example: $S = A, S_c = A_c = Q^\top A Q$

Coarsening, reconstruction

$$QX_c = \underbrace{QQ^+}_{RSA} X \quad \text{is approximately } X$$

Choice of coarsened propagation

Natural/previous choice: if $S = f(A)$, take $S_c = f(A_c)$

- That is, $S = A \to S_c = A_c$, $S = L \to S_c = L_c$, etc.
- Basically ignores that the small graph ***comes from a coarsening process***

Does not work! Example: $S = A, S_c = A_c = Q^\top AQ$

Coarsening, reconstruction

$$QX_c = \underbrace{QQ^+}_{RSA} X \quad \text{is approximately } X$$

Coarsening, ***propagation,*** *reconstruction*

$$QA_cX_c = \underbrace{QQ^\top}_{???} A \underbrace{QQ^+}_{RSA} X \quad \text{is } \textbf{\textit{not}} \text{ approximately } AX$$

Immediate fix!

Simple fix! $\boxed{S_c = Q^+ S Q}$

Immediate fix!

Simple fix! $\boxed{S_c = Q^+ S Q}$

Then $$QS_cX_c = \underbrace{QQ^+}_{RSA} S \underbrace{QQ^+}_{RSA} X \approx SX$$

Immediate fix!

Simple fix! $\boxed{S_c = Q^+ S Q}$ Then $QS_cX_c = \underbrace{QQ^+}_{RSA} S \underbrace{QQ^+}_{RSA} X \approx SX$

Can be transferred to guarantees on GNN training under appropriate assumptions...

Assumptions

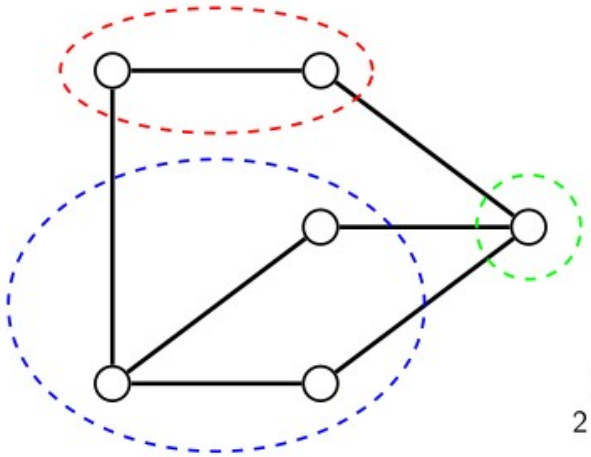
- There is a constant C_J such that $|J(x) - J(x')| \leq C_J \|x - x'\|_L$, with J the loss function
- σ is $\mathcal{R}$-preserving, that is, for all $x \in \mathcal{R}$, we have $\sigma(x) \in \mathcal{R}$, $\|\sigma(x) - \sigma(x')\|_L \leq C_\sigma \|x - x'\|_L$, σ and Q^+ commute: $\sigma(Q^+ y) = Q^+ \sigma(y)$.

For all node features $X \in \mathbb{R}^{N \times d}$ such that $X_{:,i} \in \mathcal{R}$, denoting by $\theta^\star = \arg\min_{\theta \in \Theta} R(\theta)$ and $\theta_c = \arg\min_{\theta \in \Theta} R_c(\theta)$, we have

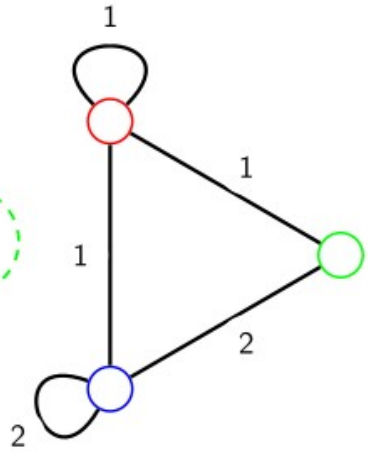
$$\boxed{R(\theta_c) - R(\theta^\star) \leq C \epsilon_{L,Q,\mathcal{R}} \|X\|_{:,L}}$$

Properties

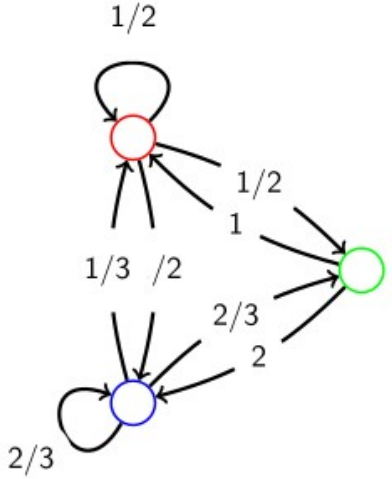
$$S_c = Q^+ S Q$$



$S = A$



$S_c = A_c$

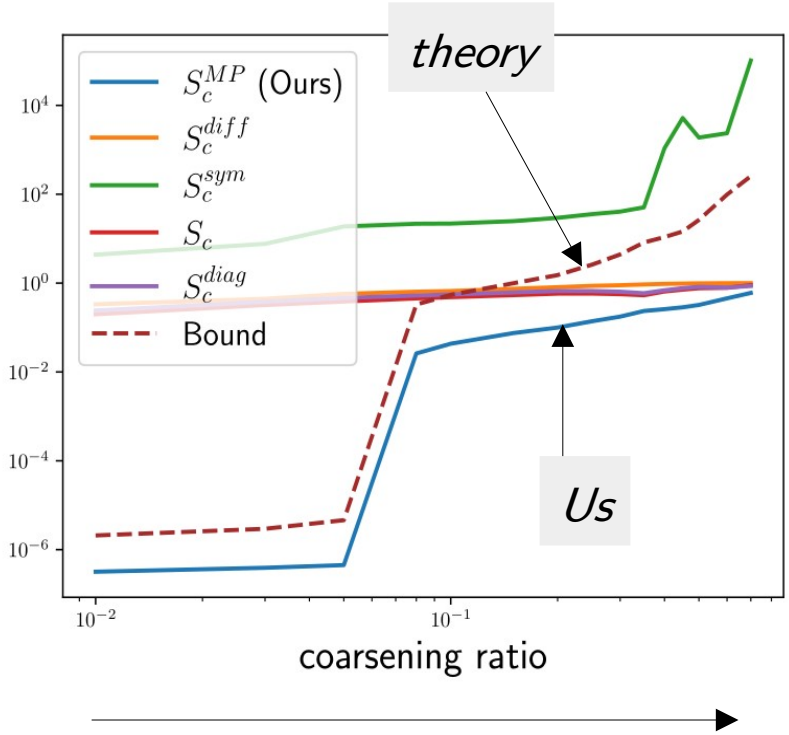


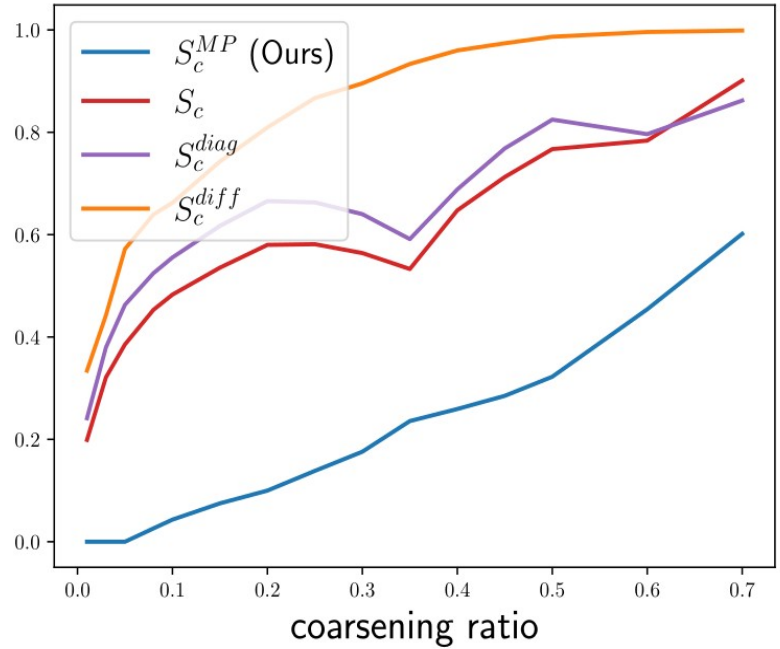
$S_c = Q^+ A Q$

- ***Not symmetric*** in general!
- Twice the memory footprint (mild)
- Can be applied to any propagation matrix!

Numerical results

Approx error





Smaller graph

Numerical Results

Dataset	# Nodes	# Edges	# Features	#classes
Reddit	232,965	114,615,892	602	41
Reddit90	23,298	8,642,864	602	41
Reddit99	2,331	10,838	602	41
Cora PCC	2,485	10,138	1,433	7
Cora70	746	3,716	1,433	7
Citeseer PCC	2,120	7,358	3,703	6
Citeseer70	636	2,122	3,703	6

SGC	Cora		Citeseer		Reddit	
r	0.5	0.7	0.5	0.7	0.9	0.99
S_c^{sym}	16.1 ± 3.8	16.4 ± 4.7	18.6 ± 4.6	19.8 ± 5.0	37.1 ± 6.6	3.7 ± 5.5
S_c^{diff}	21.8 ± 2.2	13.6 ± 2.8	30.5 ± 0.2	23.1 ± 0.0	18.3 ± 0.0	14.9 ± 0.0
S_c	78.7 ± 0.0	74.6 ± 0.1	72.8 ± 0.1	72.5 ± 0.1	87.5 ± 0.1	37.3 ± 0.0
S_c^{diag}	78.7 ± 0.1	77.3 ± 0.0	73.4 ± 0.1	73.1 ± 0.4	87.6 ± 0.1	37.3 ± 0.0
S_c^{MP} **(ours)**	**80.3** ± 0.1	**78.5** ± 0.0	**74.6** ± 0.1	**74.2** ± 0.1	**90.2** ± 0.0	**64.1** ± 0.0
Full Graph	81.6 ± 0.1		73.6 ± 0.0		94.9	

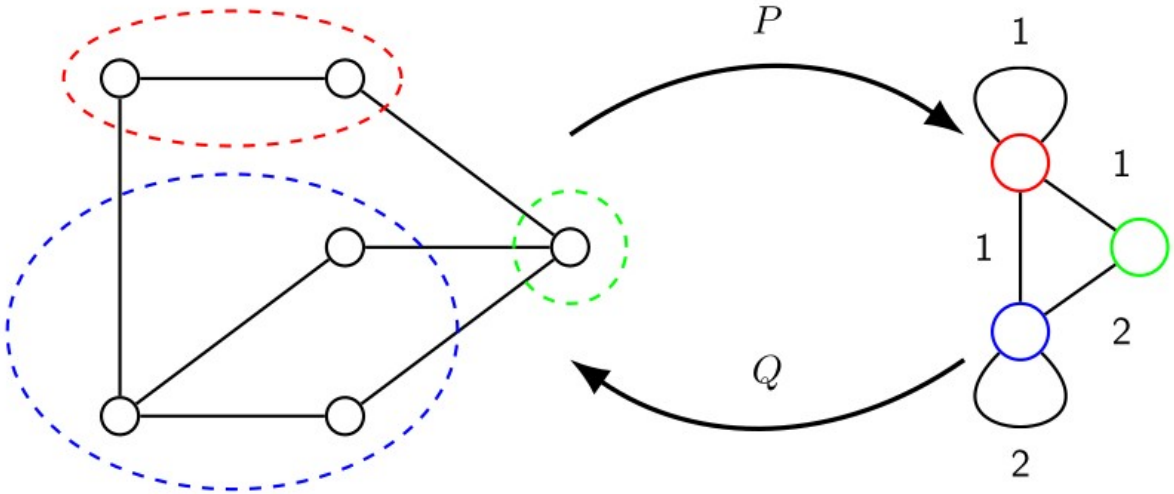
GCNconv	Cora		Citeseer		Reddit	
r	0.5	0.7	0.5	0.7	0.9	0.99
S_c^{sym}	78.1 ± 1.3	30.8 ± 2.5	62.5 ± 11	52.7 ± 3.6	48.1 ± 8.9	34.8 ± 4.0
S_c^{diff}	74.5 ± 0.9	62.6 ± 7.1	71.2 ± 1.7	37.6 ± 0.9	71.3 ± 1.0	18.7 ± 1.7
S_c	79.9 ± 0.9	78.1 ± 1.0	70.7 ± 1.0	67.1 ± 3.1	88.0 ± 0.1	54.2 ± 2.4
S_c^{diag}	**80.4** ± 0.8	**78.6** ± 1.3	70.2 ± 0.8	69.3 ± 1.9	**88.1** ± 0.2	55.5 ± 1.8
S_c^{MP} **(ours)**	79.8 ± 1.5	78.2 ± 0.9	**72.0** ± 0.8	**70.0** ± 1.0	84.4 ± 0.3	**60.3** ± 0.9
Full Graph	81.6 ± 0.6		73.1 ± 1.5		OOM	

-> difficult to treat ***very large graphs***, as the RSA-coarsening itself is costly... (future work)

Outline

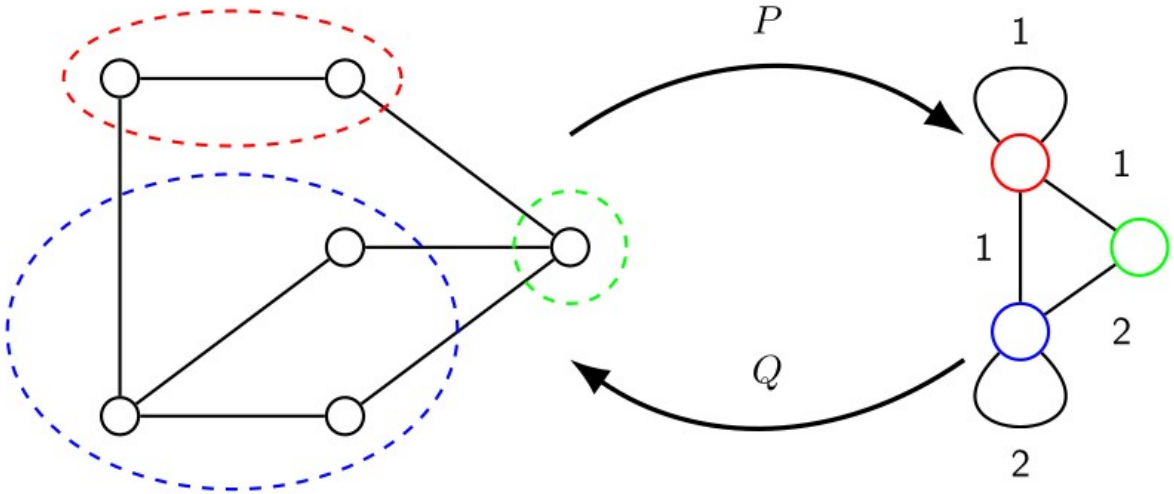
1. Graph coarsening...
2. ... with Graph Neural Networks
3. Bonus: taxonomy of graph coarsening

Reduction vs lifting



- Until now, $P = Q^+$

Reduction vs lifting



- Until now, $P = Q^+$
- But only Q is important for consistency!
- **Can we change P?**

Lemma. (Consistency of Laplacian [1])
Let Q be a well-partitioned matrix. The two following properties are equivalent:

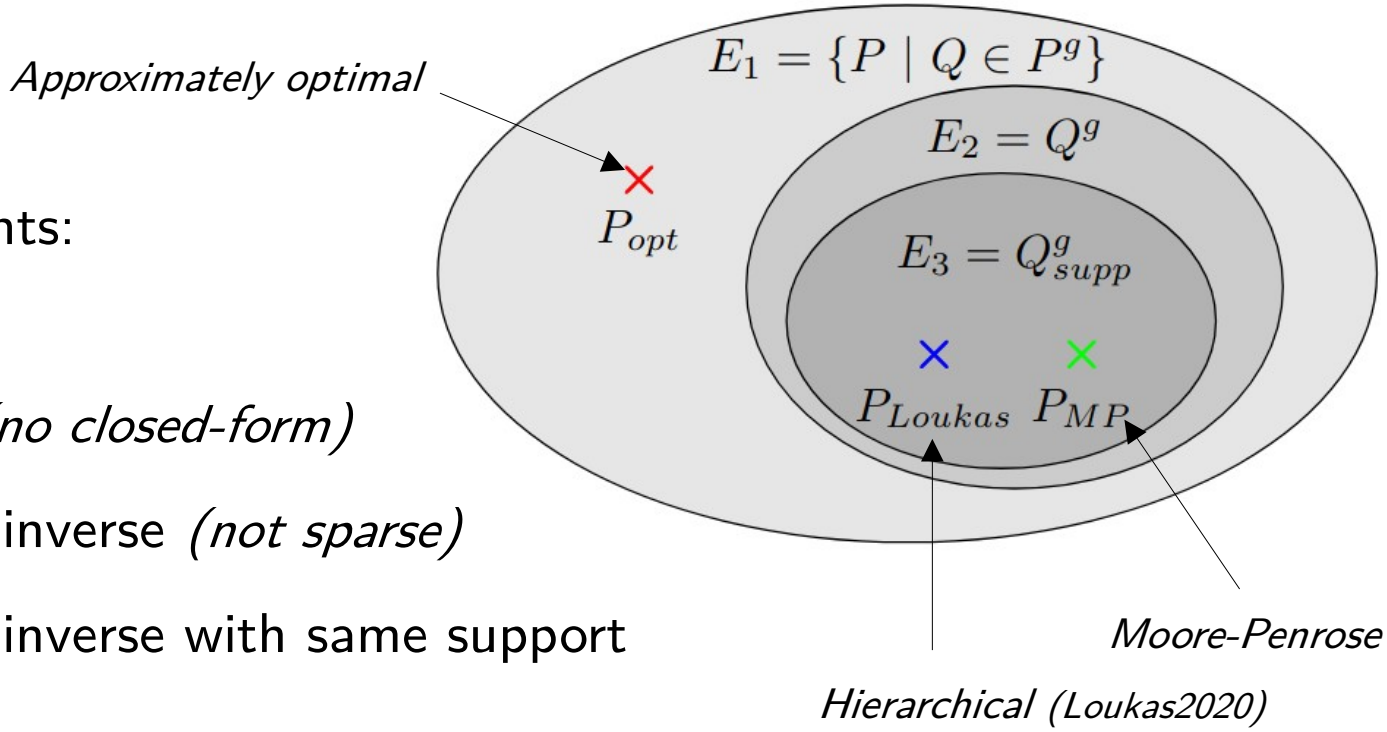
$$Q \text{ is proportional to a binary matrix.}$$
$$\Updownarrow$$
$$\forall A, \quad L(A_c) = Q^\top L Q$$

[1] Andreas Loukas, *Graph Reduction with Spectral and Cut Guarantees*, JMLR 2019.

Several choices for P

Increasing level of constraints:

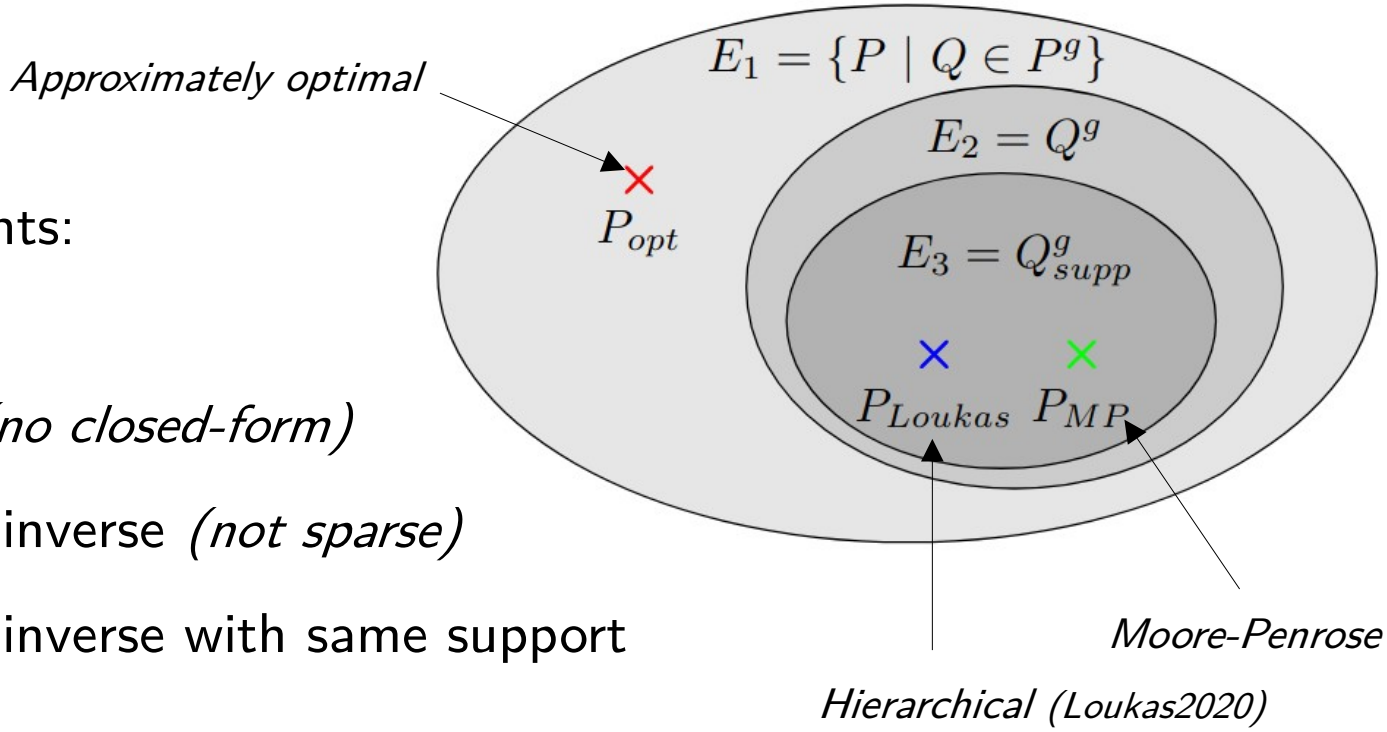
- E1: Generalized inverse *(no closed-form)*
- E2: Generalized reflexive inverse *(not sparse)*
- E3: Generalized reflexive inverse with same support



Several choices for P

Increasing level of constraints:

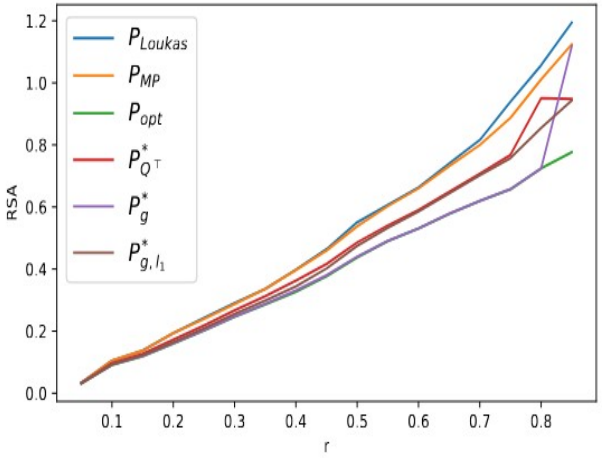
- E1: Generalized inverse *(no closed-form)*
- E2: Generalized reflexive inverse *(not sparse)*
- E3: Generalized reflexive inverse with same support



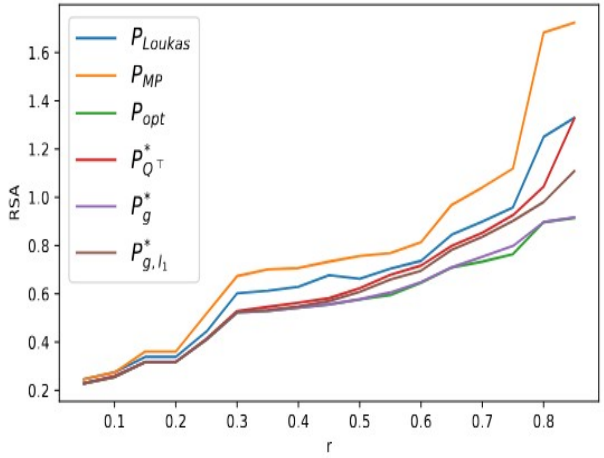
-> closed-form (E2 and 3) can be used to ***optimize*** the RSA!

Several choices for P

-> can be used to ***further optimize*** the RSA!



(a) Cora graph, combinatorial Laplacian $\mathcal{L}$



(b) Cora graph, self-loop normalized Laplacian L

SGC	Cora			Citeseer		
r	0.3	0.5	0.7	0.3	0.5	0.7
P_{Loukas}	80.5 ± 0.0	79.7 ± 0.0	76.8 ± 0.0	72.6 ± 0.3	71.7 ± 0.1	**69.7** ± 0.7
P_{MP}	80.5 ± 0.0	**80.1** ± 0.0	77.7 ± 0.0	72.8 ± 0.5	72.7 ± 0.0	69.5 ± 0.3
P_{opt}	77.1 ± 0.6	75.9 ± 0.1	73.8 ± 0.3	70.9 ± 0.2	70.2 ± 0.1	67.3 ± 0.4
$P^*_{Q^\top}$	80.3 ± 0.0	80.0 ± 0.1	77.2 ± 0.0	72.7 ± 0.3	72.6 ± 0.5	67.6 ± 0.2
P^*_g	**80.7** ± 0.0	80.0 ± 0.0	77.6 ± 0.0	72.6 ± 0.2	**72.7** ± 0.0	68.6 ± 0.4
P^*_{g,l_1}	80.4 ± 0.0	79.2 ± 0.0	**78.3** ± 0.0	**73.0** ± 0.0	71.2 ± 0.1	69.2 ± 0.4
Full Graph		81.0 ± 0.1			71.6 ± 0.1	

GCN	Cora			Citeseer		
r	0.3	0.5	0.7	0.3	0.5	0.7
P_{Loukas}	**80.6** ± 0.8	80.5 ± 1.0	78.1 ± 1.4	71.0 ± 1.6	72.2 ± 0.6	70.4 ± 0.8
P_{MP}	80.4 ± 1.0	80.7 ± 0.9	78.6 ± 0.9	70.8 ± 1.9	72.1 ± 1.0	**71.0** ± 1.0
P_{opt}	73.7 ± 1.5	63.3 ± 1.4	55.11 ± 2.4	64.6 ± 0.7	50.4 ± 1.6	42.6 ± 4.0
$P^*_{Q^\top}$	80.5 ± 0.9	80.9 ± 0.6	78.0 ± 0.9	**71.1** ± 1.5	**72.3** ± 0.7	70.0 ± 0.9
P^*_g	**80.6** ± 1.1	**81.3** ± 0.6	**78.7** ± 0.9	**71.1** ± 1.7	72.1 ± 1.2	69.6 ± 1.0
P^*_{g,l_1}	80.4 ± 0.9	80.0 ± 0.9	78.2 ± 0.7	70.2 ± 1.8	66.8 ± 1.1	66.7 ± 1.2
Full Graph		81.3 ± 0.8			70.9 ± 1.4	

Can be used within GNNs

Outline

Conclusion

Conclusion

- ***Graph reduction*** is unavoidable in Graph Machine Learning, but still in its infancy
- Both **coarsening** and **sampling** have pros and cons
- Graph coarsening is *difficult/costly*, but bears many promising directions
  - pooling CNN-like? **Heterogeneous graphs?**

Antonin Joly, Nicolas Keriven, Aline Roumy. **Graph Coarsening with Message-Passing Guarantees** . *NeurIPS 2024*.

Antonin Joly, Nicolas Keriven, Aline Roumy. **Taxonomy of reduction matrices for Graph Coarsening**. *NeurIPS 2025*.

European Research Council
Established by the European Commission

FRANCE 2030
PROGRAMME DE RECHERCHE
INTELLIGENCE ARTIFICIELLE