Partition and Code:

Learning how to Compress Graphs

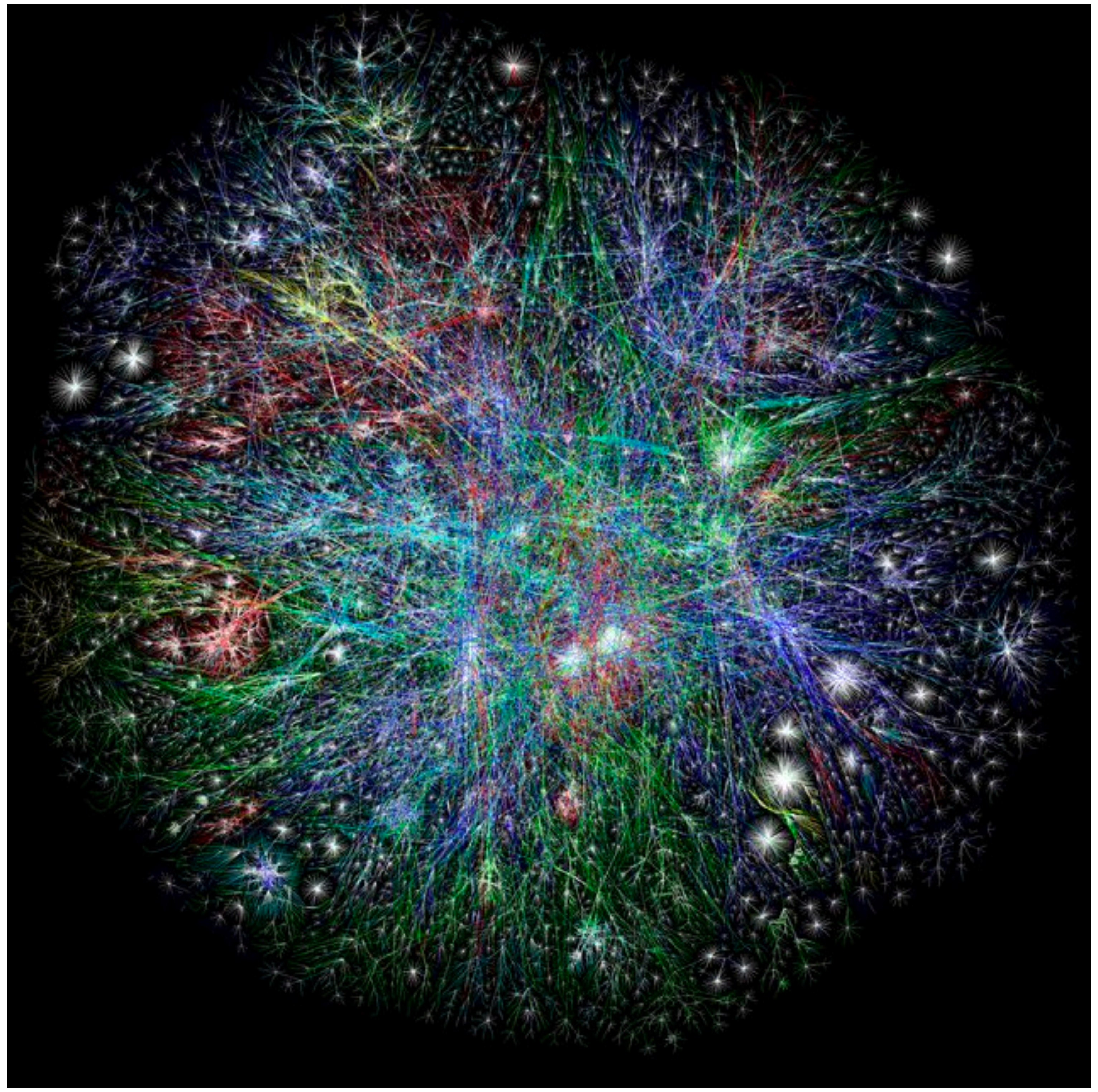
Giorgos Bouritsas, Andreas Loukas, Nikolaos Karalias, Michael Bronstein

Imperial College London

EPFL

Why do we care about compressing graphs?

Single graph



Internet
https://www.opte.org/

Practical

transmission/storage/processing

Graph datasets

dataset	number of graphs	size (processed)
ogbg-ppa	158K	30GB
ogbg-code	453K	3GB
ogbg-molpcba	438K	2GB

Theoretical

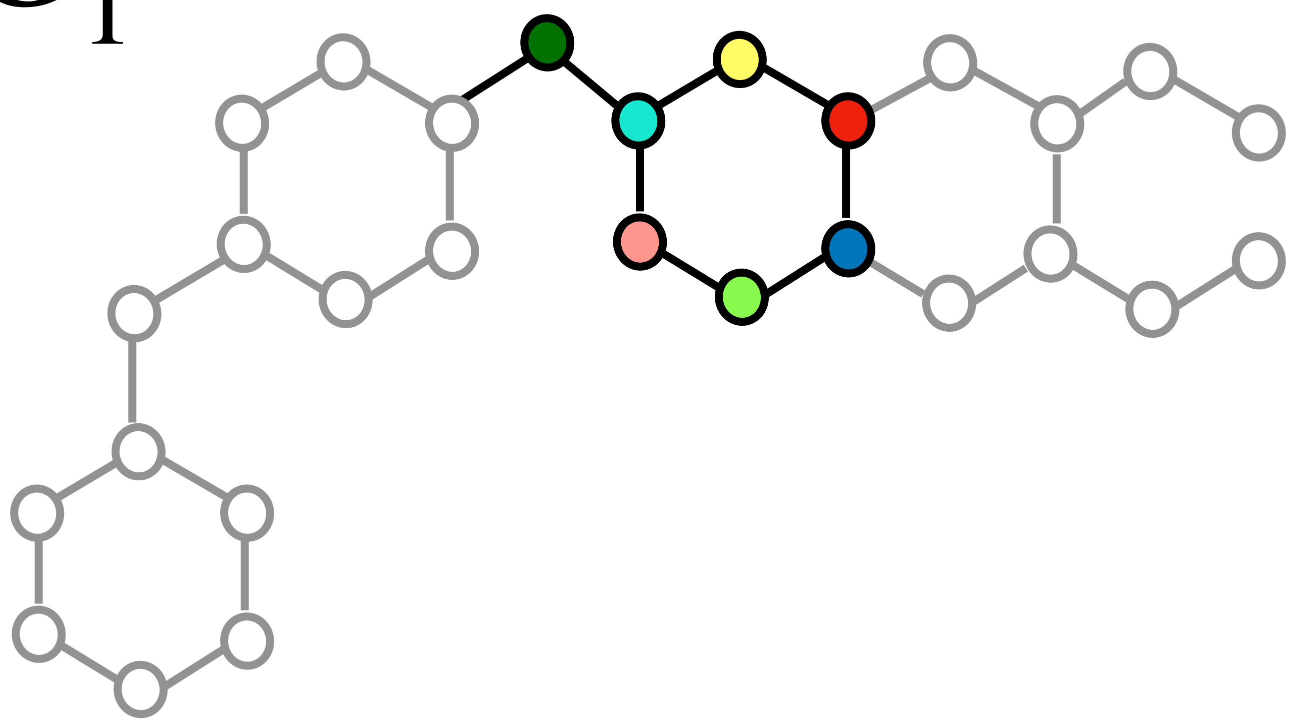
Fundamental problem in computer science

What is the role of machine learning in compression?

Lossless Graph Compression

Source p

G_1



Encoder
(Description Method)

CODE

...00001111010101011010...

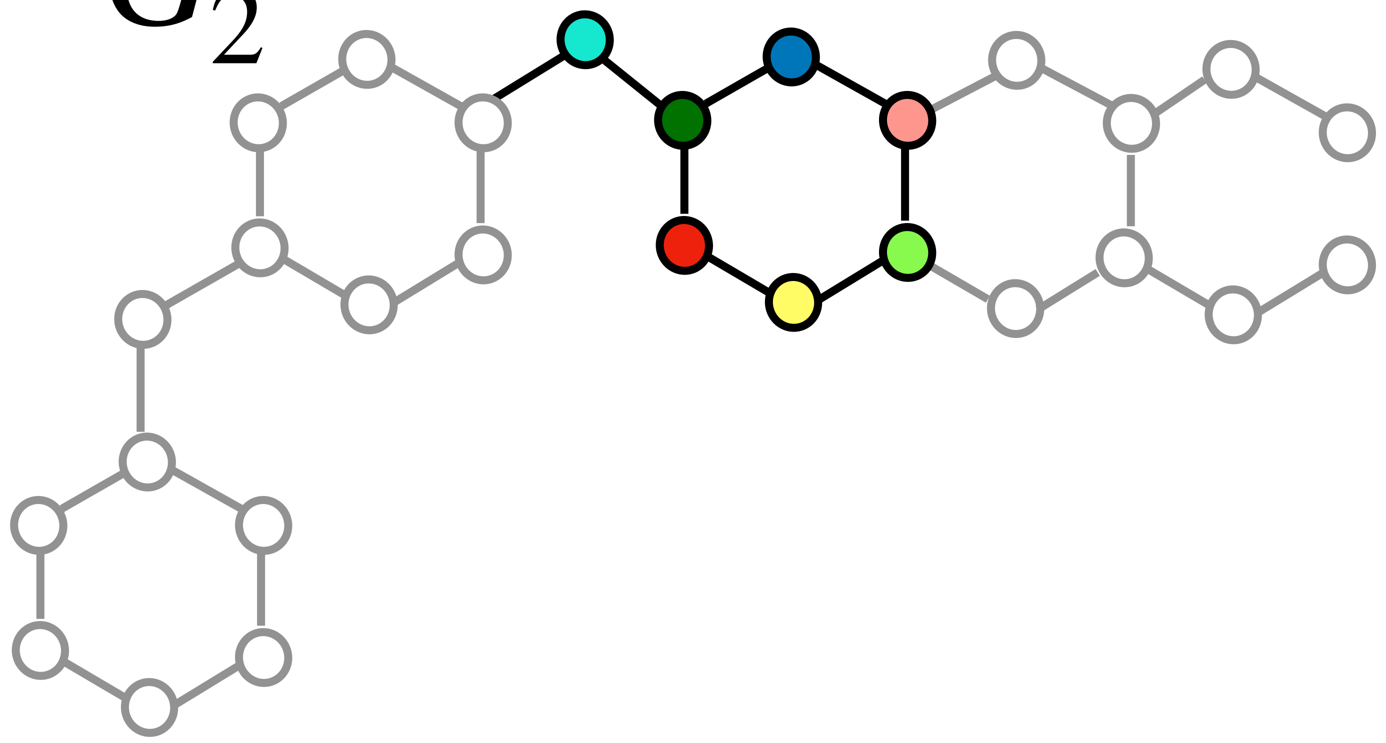
$\underbrace{\hspace{3cm}}$

Description length in bits $\mathrm{L}_{\text{CODE}}(G)$

Decoder

CODE^{-1}

G_2



$$G_1 \cong G_2$$

***This work*: sample space of isomorphism classes**
***Previous work*: (1) labelled graphs or (2) a single large graph**

Information theory basics

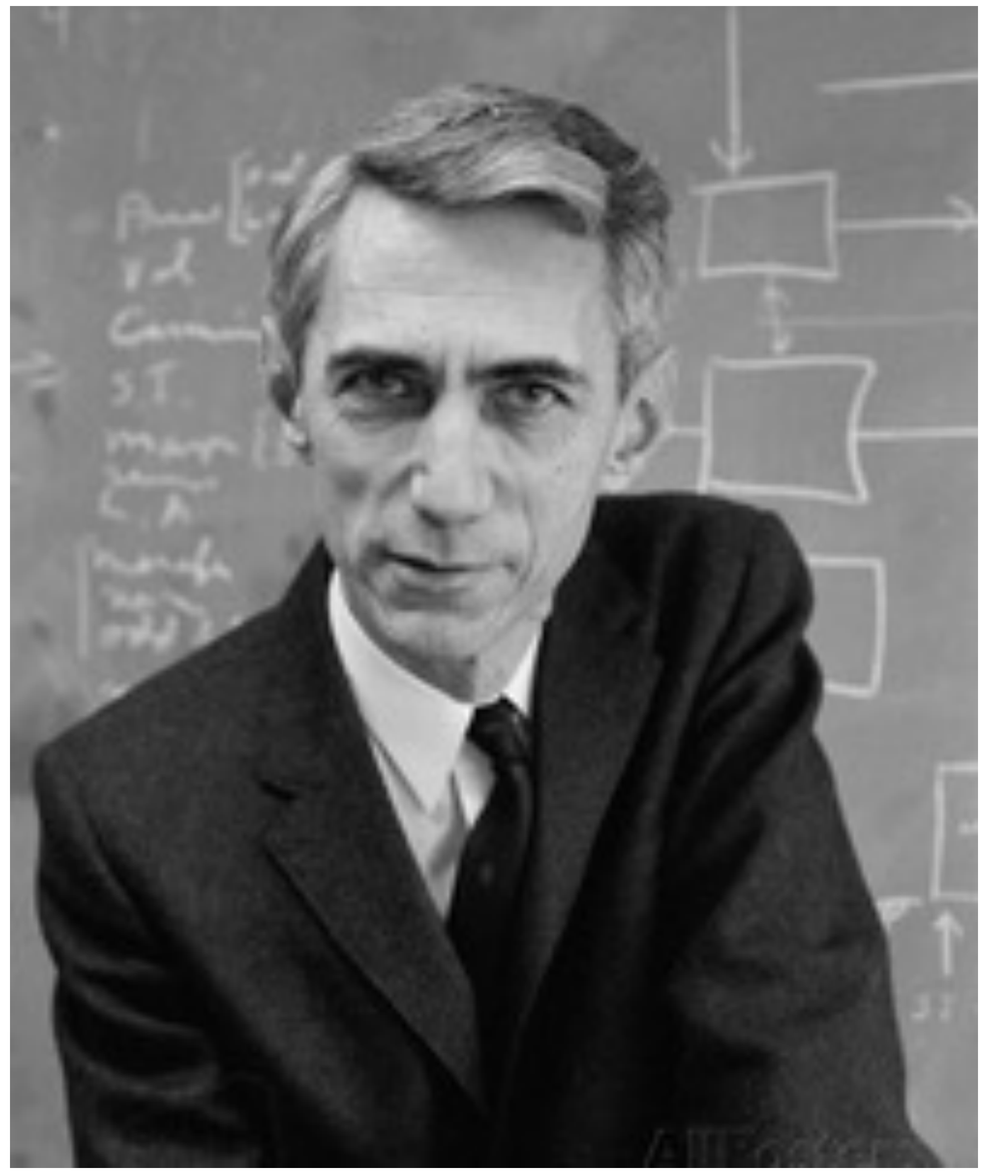
Objective: find a description method that minimises the expected description length.

$$\min_{\text{CODE}} \mathbb{E}_{G \sim p}[\mathrm{L}_{\text{CODE}}(G)]$$

Information theory basics

Objective: find a description method that minimises the expected description length.

$$\min_{\text{CODE}} \mathbb{E}_{G\sim p}[\mathrm{L}_{\text{CODE}}(G)]$$



Shannon's source coding theorem (informal)*: For all description methods* CODE *it holds that:*

$$\mathbb{E}_{G\sim p}[\mathrm{L}_{\text{CODE}}(G)] \geq \mathbb{E}_{G\sim p}[-\log p(G)] = \mathrm{H}_{G\sim p}[G],$$

where $\mathrm{H}_{G\sim p}[G]$ is the *Entropy* of the r.v. G.

Information theory basics

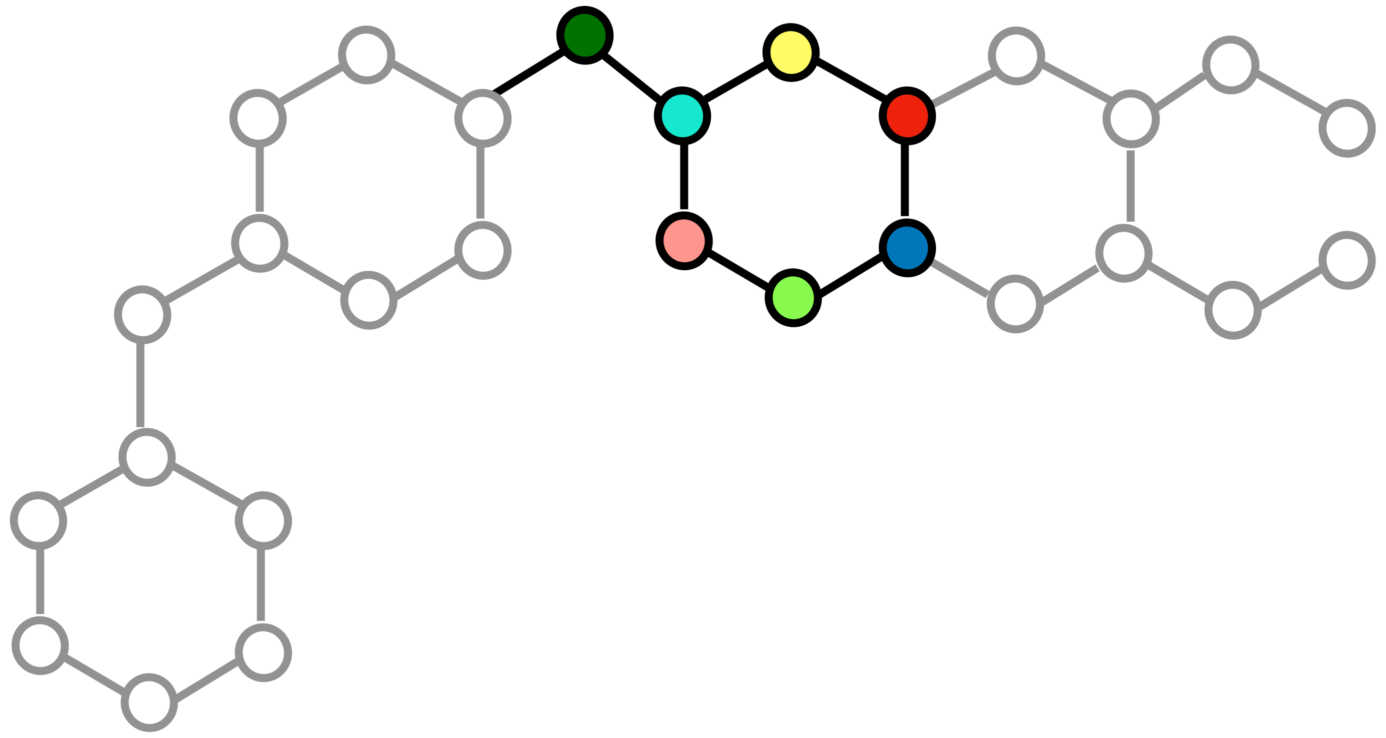
$$\mathrm{L}_{\text{CODE}}(G) = -\log p(G) \Leftrightarrow \text{CODE is optimal.}$$

- Optimise for **probability distributions** instead of description methods.

$$\min_q \mathbb{E}_{G\sim p}[-\log q(G)]$$

- Every distribution q on a finite sample space can be converted to a uniquely decodable code using an entropy coder (e.g., Arithmetic Coding, ANS).

Warmup: Simple uninformative encodings



Adjacency matrix $\xrightarrow{\textit{Encoder}}$

0	1	0	0	0	0	0
1	0	1	0	0	0	1
0	1	0	1	0	0	0
0	0	1	0	1	0	0
0	0	0	1	0	1	0
0	0	0	0	1	0	1
0	1	0	0	0	1	0

C = …01000001010001…

$$\mathrm{L}_C(G) = n^2 \Leftrightarrow q(G) = \frac{1}{2^{n^2}}$$

Erdős - Rényi $\xrightarrow{\textit{Encoder}}$

m = 14

$\mathscr{E}$ = {(1,2), (2,1), (2,3), (2,7), (3,2), (3,4), (4,3), (4,5), (5,4), (5,6), (6,5), (6,7), (7,2), (7,6)}

C = …101000111000…

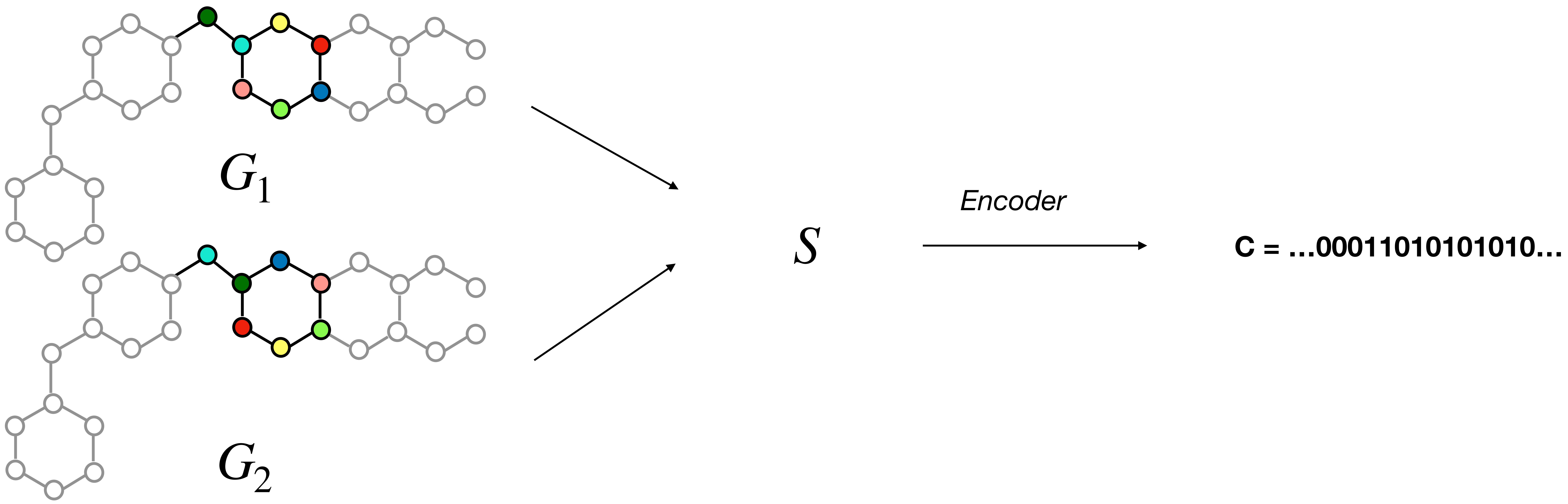
$$\mathrm{L}_C(G) = \log(n^2+1) + \log\binom{n^2}{m}$$

$$\Updownarrow$$

$$q(G) = \frac{1}{n^2+1}\frac{1}{\binom{n^2}{m}}$$

Challenge 1: The usual suspect - Isomorphism

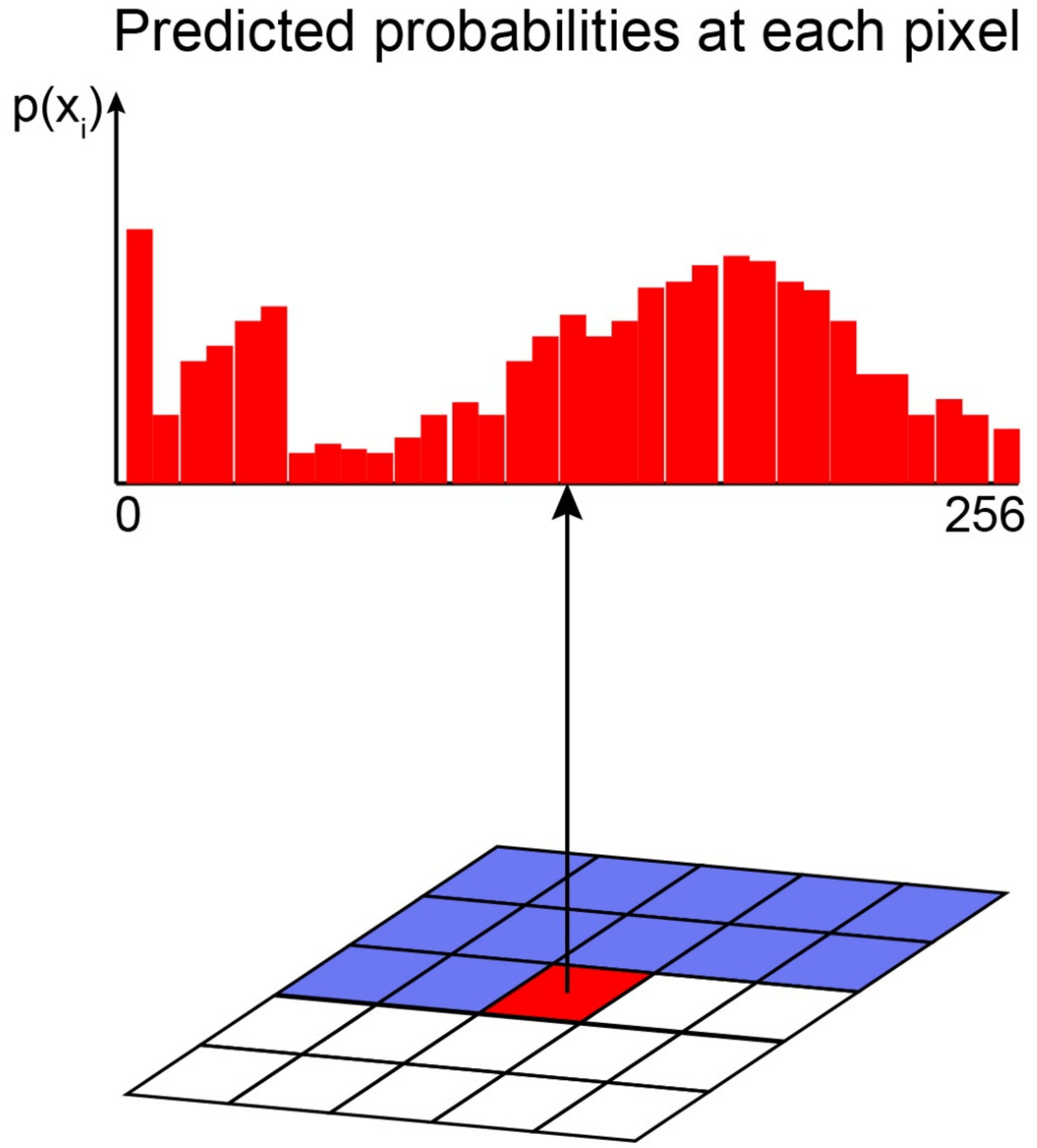
- Unique to graphs, makes the problem fundamentally different.
- To achieve optimality, the distribution needs to be defined on isomorphism classes.
- **Requires solving graph isomorphism.**



Challenge 2: Estimating & evaluating the likelihood

- Evaluate the probability **everywhere,** e.g., by decomposing the probability distribution (autoregressive models for ordered data - images and text).

$$q(X) = q(x_1)\prod_{i=2}^{n} q(x_i \,|\, x_{i-1}, x_{i-2}, \ldots x_1)$$

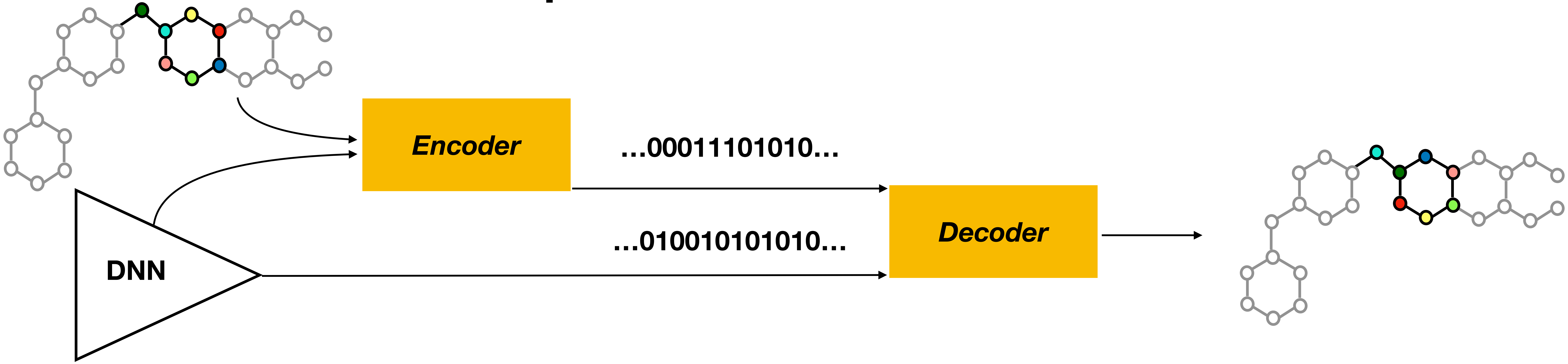


https://towardsdatascience.com/autoregressive-models-pixelcnn-e30734ede0c1

- Observation space of graphs is huge - grows with $\Omega\left(\frac{2^{n^2}}{n!}\right)$.

- **How to decompose the distribution in the absence of ordering?**

Challenge 3: The description length of the model
Compression vs Generative models



- In case of parametric model (e.g., a DNN), the encoder and the decoder need to both possess the paremetrisation of the distribution. Hence:

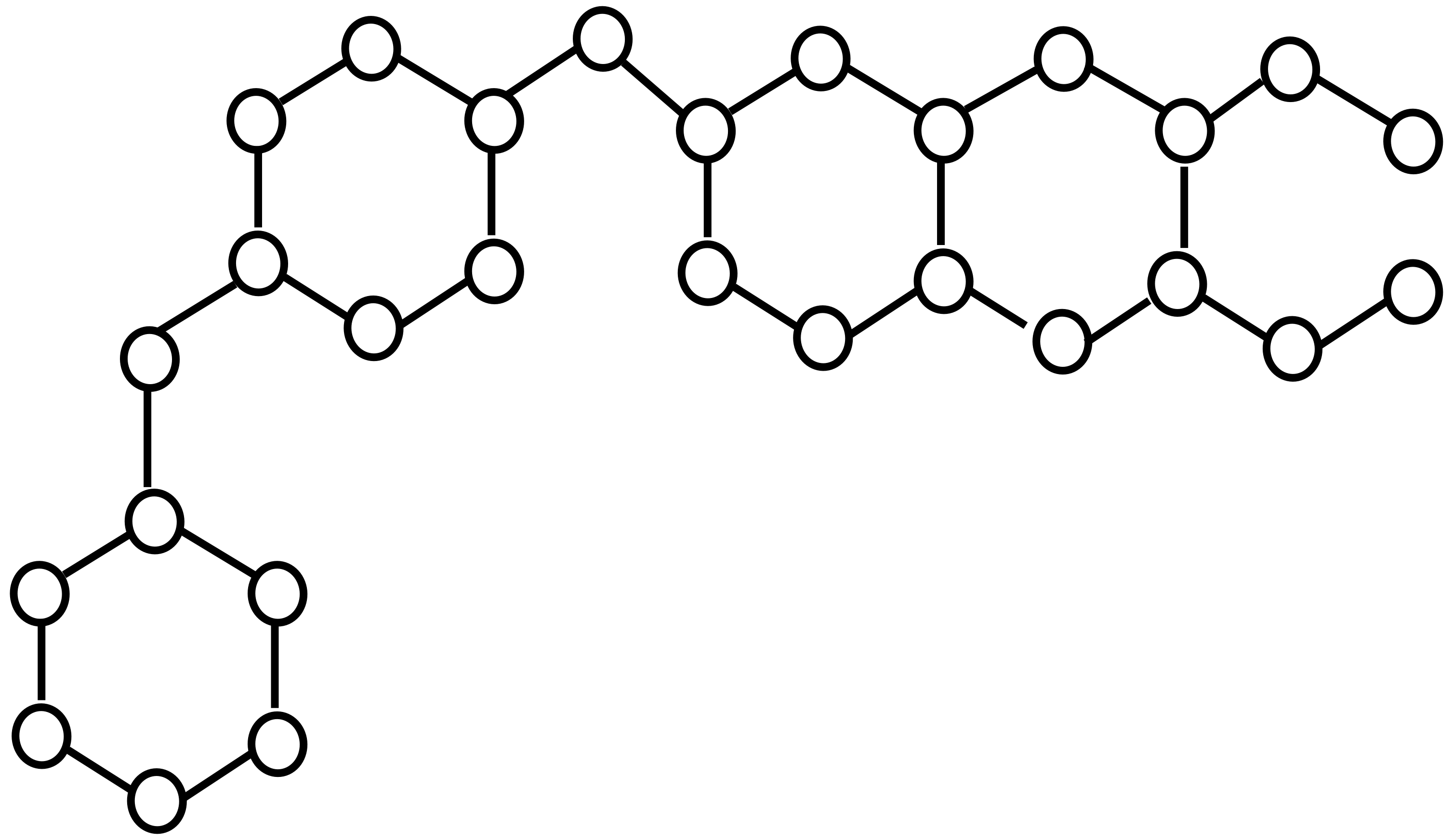
Minimum Description Length

$$\min_{q,M} \mathbb{E}_{G\sim p}[-\log q(G\,|\,M)] + \frac{1}{N}\mathrm{L}(M)$$

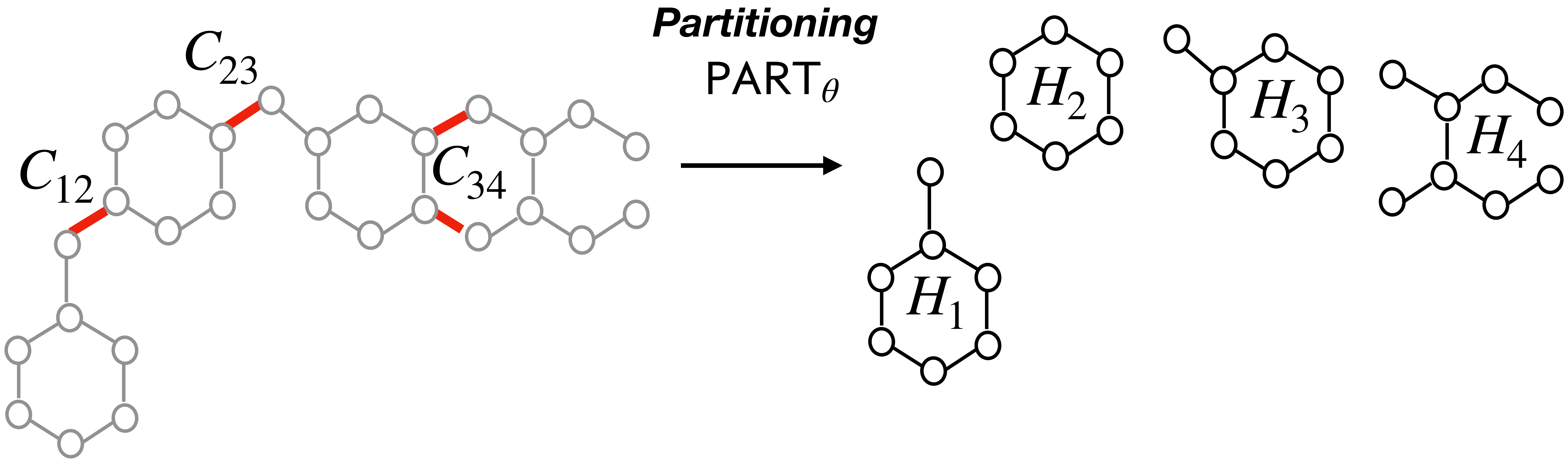
- Overparametrisation might be problematic.

- **Typically neural compressors are only optimised w.r.t. the cross-entropy.**

Partition and Code: Pipeline



Partition and Code: Pipeline



Cuts

$C = \{C_{12}, C_{13}, C_{14}, C_{23}, C_{24}, C_{34}\}$

Subgraphs

$\mathscr{H} = \{H_1, H_2, H_3, H_4\}$

Partition and Code: Pipeline

Cuts

$C = \{C_{12}, C_{13}, C_{14}, C_{23}, C_{24}, C_{34}\}$

Subgraphs

$\mathscr{H} = \{H_1, H_2, H_3, H_4\}$

Partition and Code: Pipeline

Partition and Code: Pipeline

Cuts

$C = \{C_{12}, C_{13}, C_{14}, C_{23}, C_{24}, C_{34}\}$

Subgraphs

$\mathscr{H} = \{H_1, H_2, H_3, H_4\}$

Graph encoding: $\mathrm{L}(G \,|\, M)$

Cut Encoding: $-\log q(C \,|\, \mathscr{H})$

+

Subgraph Encoding: $-\log q(\mathscr{H} \,|\, D)$

Dictionary Encoding: $\mathrm{L}(M) = -\log q(D)$

....00010011......

....00010011......

....00010011......

How do we address the challenges?

☑ C1 Isomorphism: Dictionary

- We efficiently solve it for small graphs of size $k = O(1)$.
- tradeoff between expressivity and complexity.

How do we address the challenges?

☑ **C1 Isomorphism: Dictionary**

- We efficiently solve it for small graphs of size $k = O(1)$.
- tradeoff between expressivity and complexity.

☑ **C2 Evaluating the Likelihood: Partitioning**

- Provides us with a **learnable decomposition** of the probability distribution (subgraphs + cuts).

How do we address the challenges?

☑ **C1 Isomorphism: Dictionary**

- We efficiently solve it for small graphs of size $k = O(1)$.
- tradeoff between expressivity and complexity.

☑ **C2 Evaluating the Likelihood: Partitioning**

- Provides us with a **learnable decomposition** of the probability distribution (subgraphs + cuts).

☑ **C3 The DL of the model: End-to-end optimisation + Learnable Dictionary**

$$\min_{q,M} \mathbb{E}_{G\sim p}[-\log q(G\,|\,M)] + \frac{1}{N}\mathrm{L}(M) \Rightarrow \min_{\phi,D,\theta} \mathbb{E}_{G\sim p}[-\log q_\phi(\mathrm{PART}_\theta\,|\,D)] + \frac{1}{N}\mathrm{L}(D)$$

NB: PART_θ **does not need to be transmitted.**

PART_θ **does all the heavy-lifting while** ϕ **is kept small!**

Distribution & dictionary parametrisation

1. Graph Likelihood: Subgraph Encoding + Cut encoding

$$\begin{aligned} q_\phi(G\,|\,D) &= q(\mathscr{H}\,|\,D)q(C\,|\,\mathscr{H}, D) \\ &= q_\phi(b_{dict}, b_{null})q_\phi(\mathscr{H}_{dict}\,|\,b_{dict}, D)q_{null}(\mathscr{H}_{null}\,|\,b_{null})q_{null}(C\,|\,\mathscr{H}) \end{aligned}$$

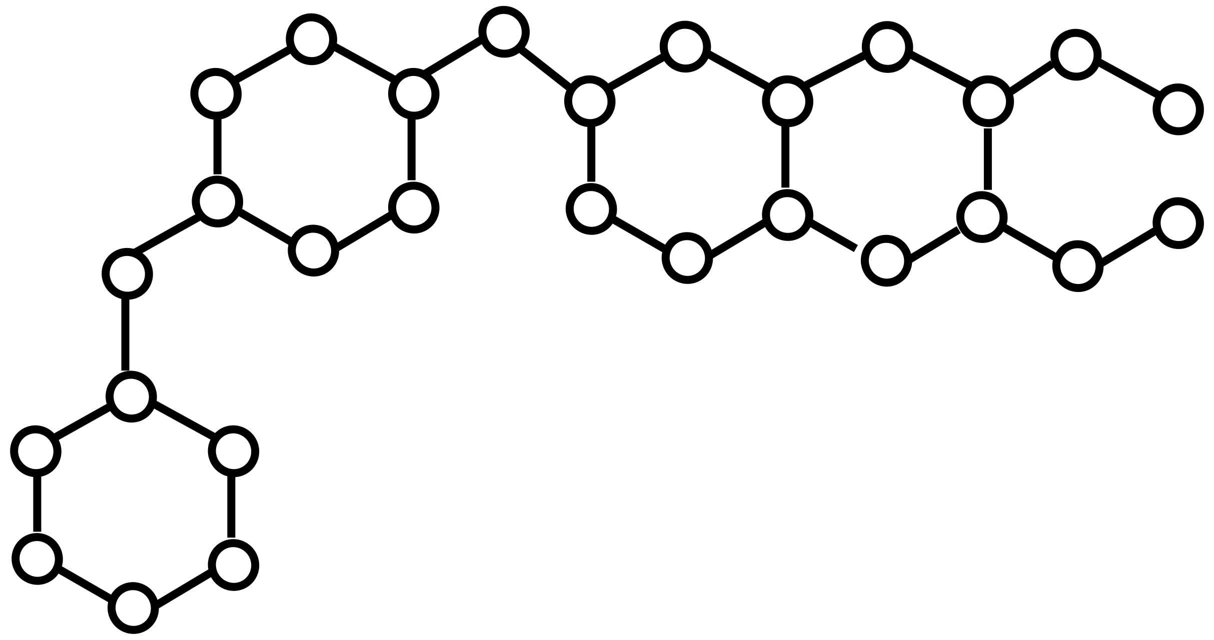
Number of subgraphs + Dictionary subgraphs + Non-dictionary subgraphs + Cuts

Parametric

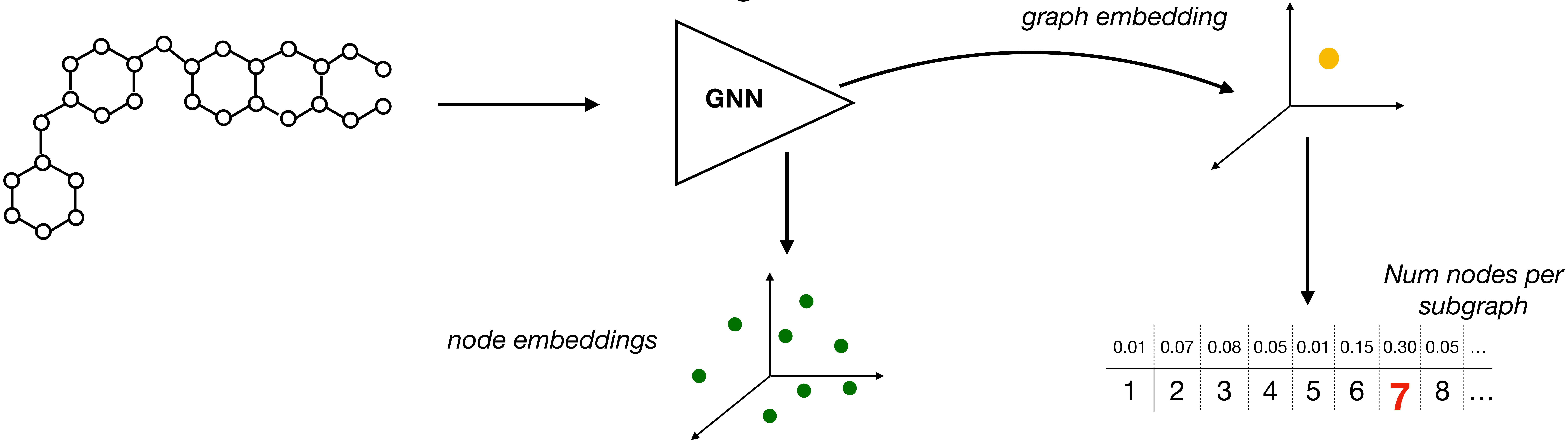
Non-parametric (null model)

***2. Dictionary DL**:* $\mathrm{L}(D) = -\sum_{a_i \in \mathfrak{U}} x_i \log q_{null}(a_i),\qquad x_i = \begin{cases} 1 & \text{if } a_i \in D \\ 0 & \text{otherwise}\,. \end{cases},\qquad$ $\mathfrak{U}$: universe

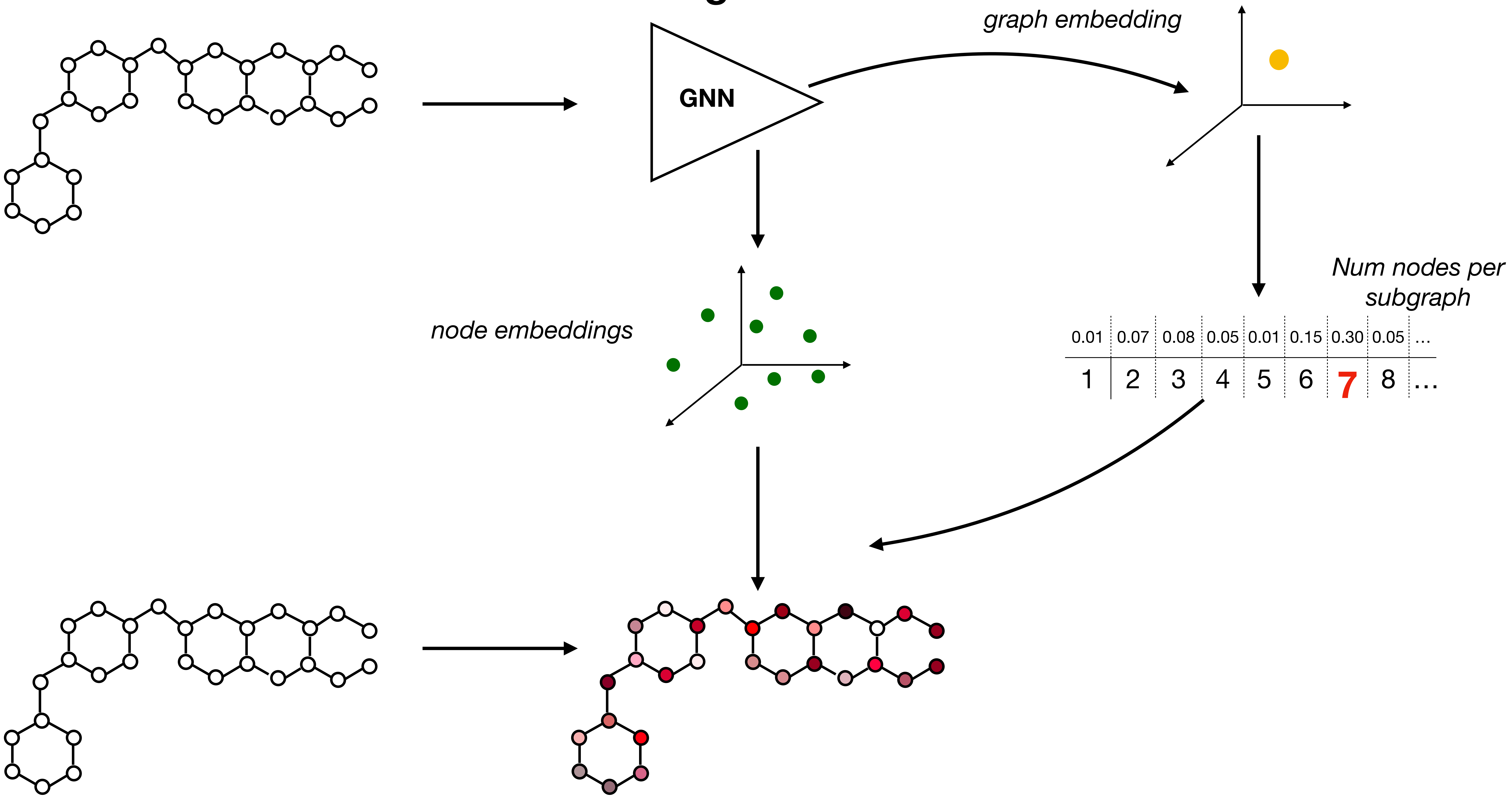
Learning to Partition



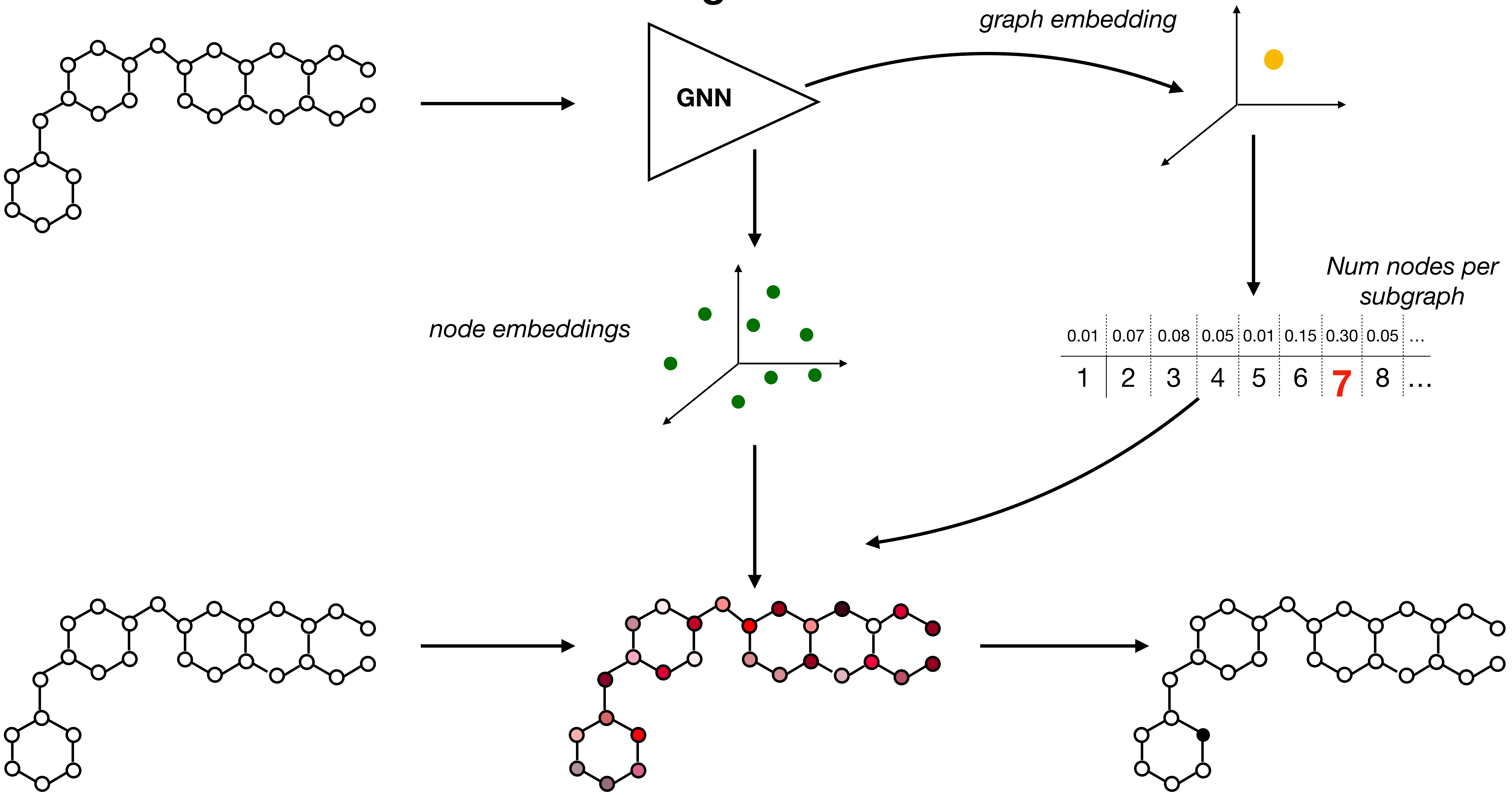
Learning to Partition



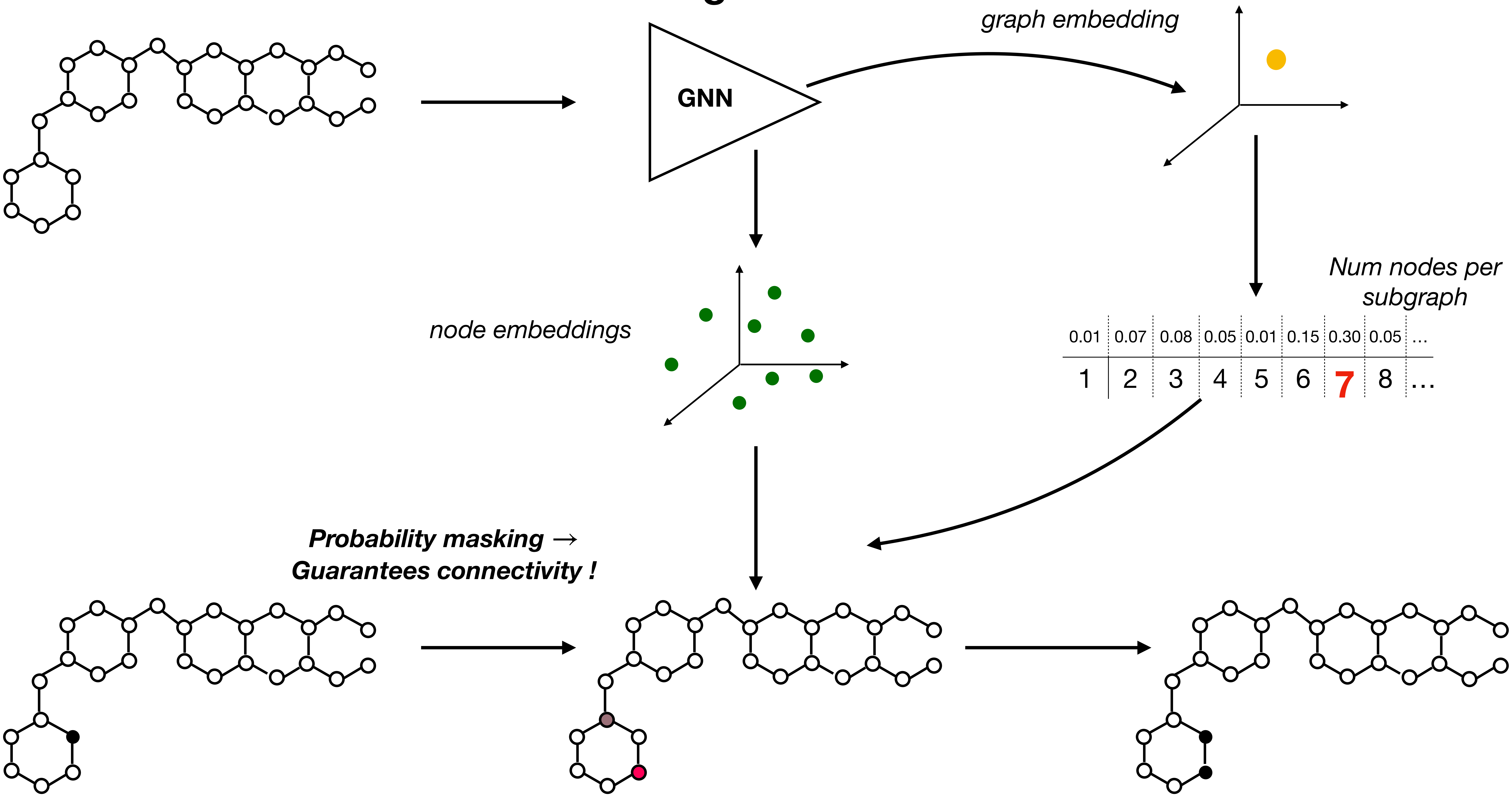
Learning to Partition



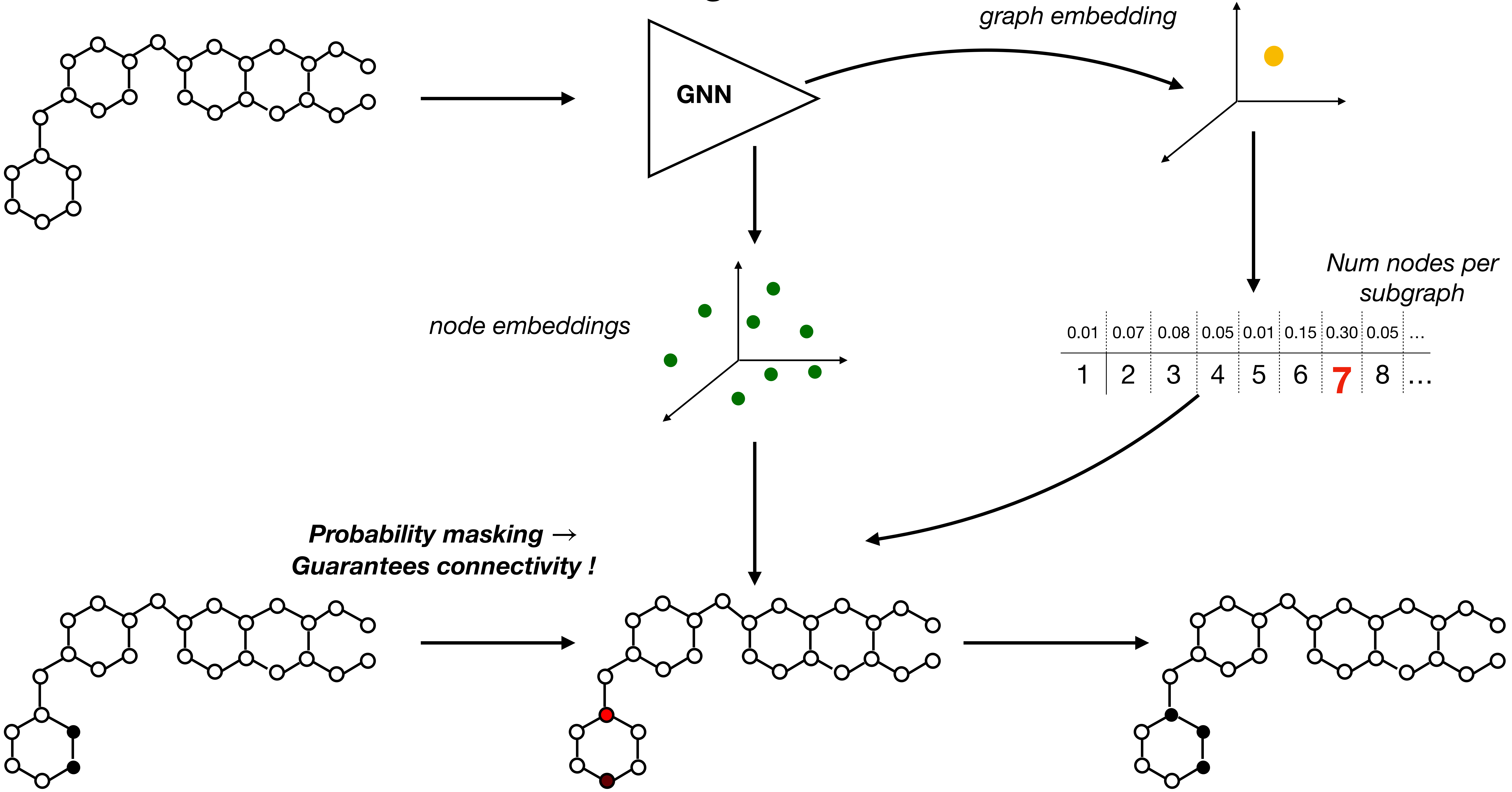
Learning to Partition



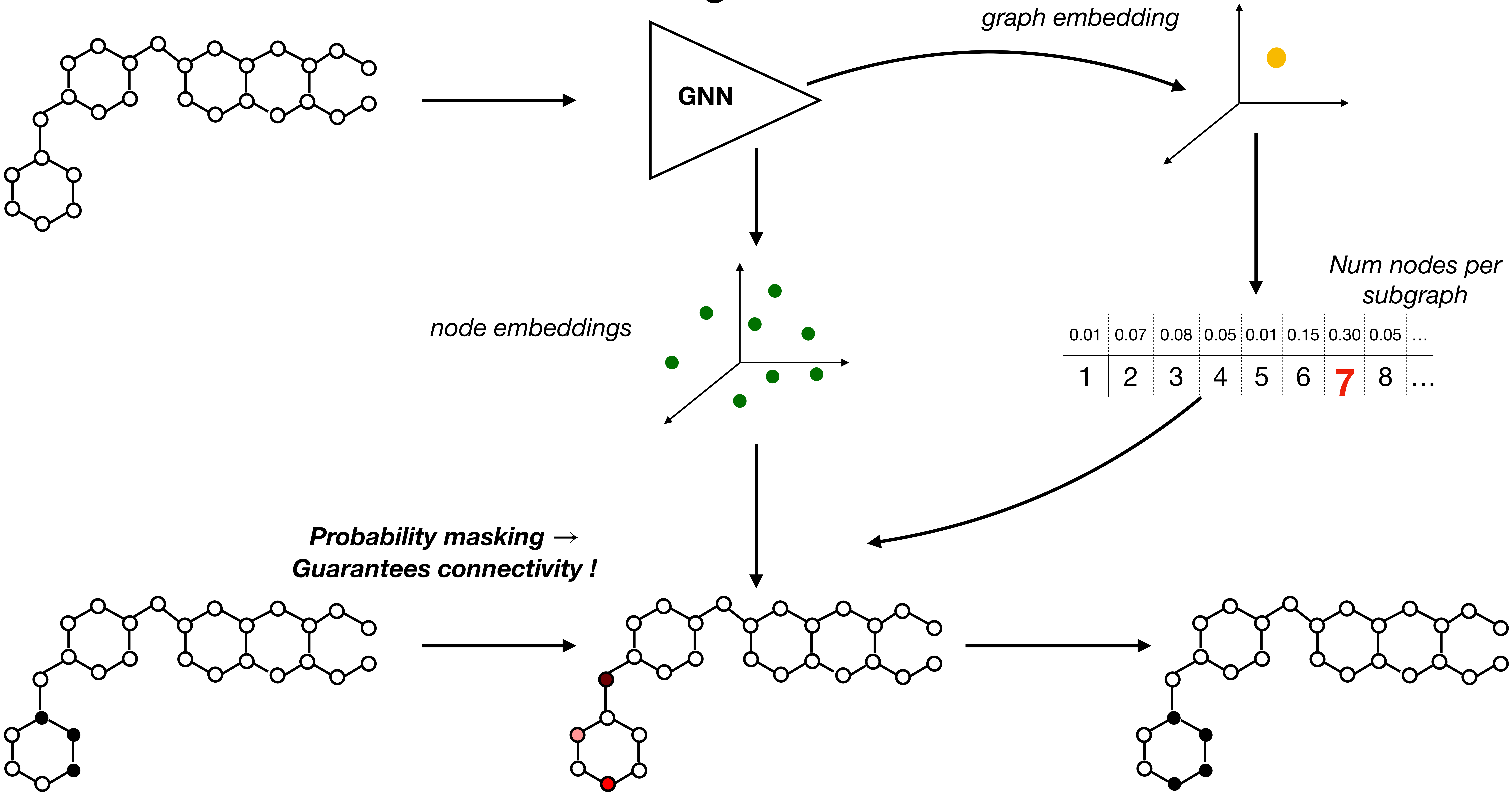
Learning to Partition



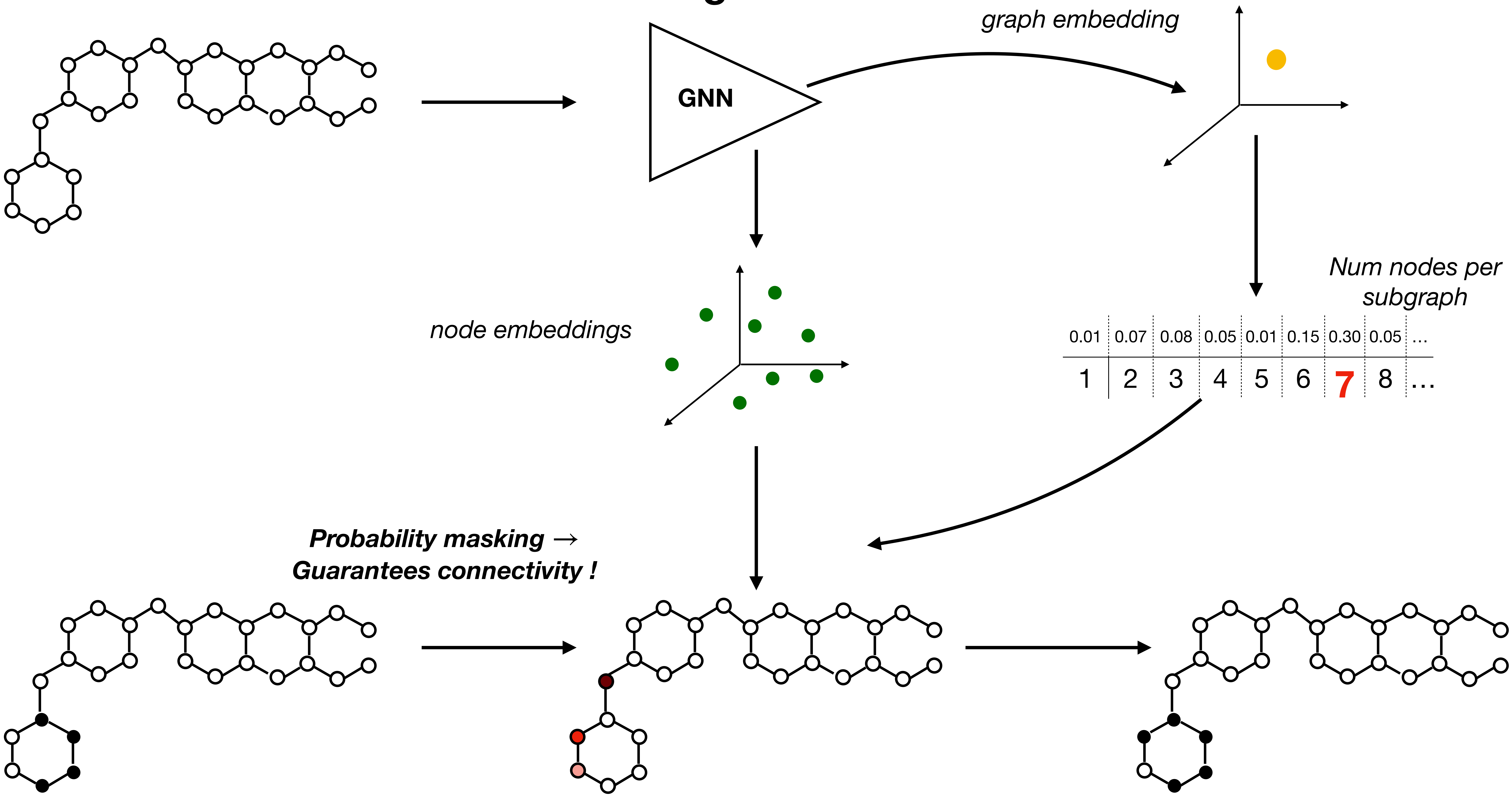
Learning to Partition



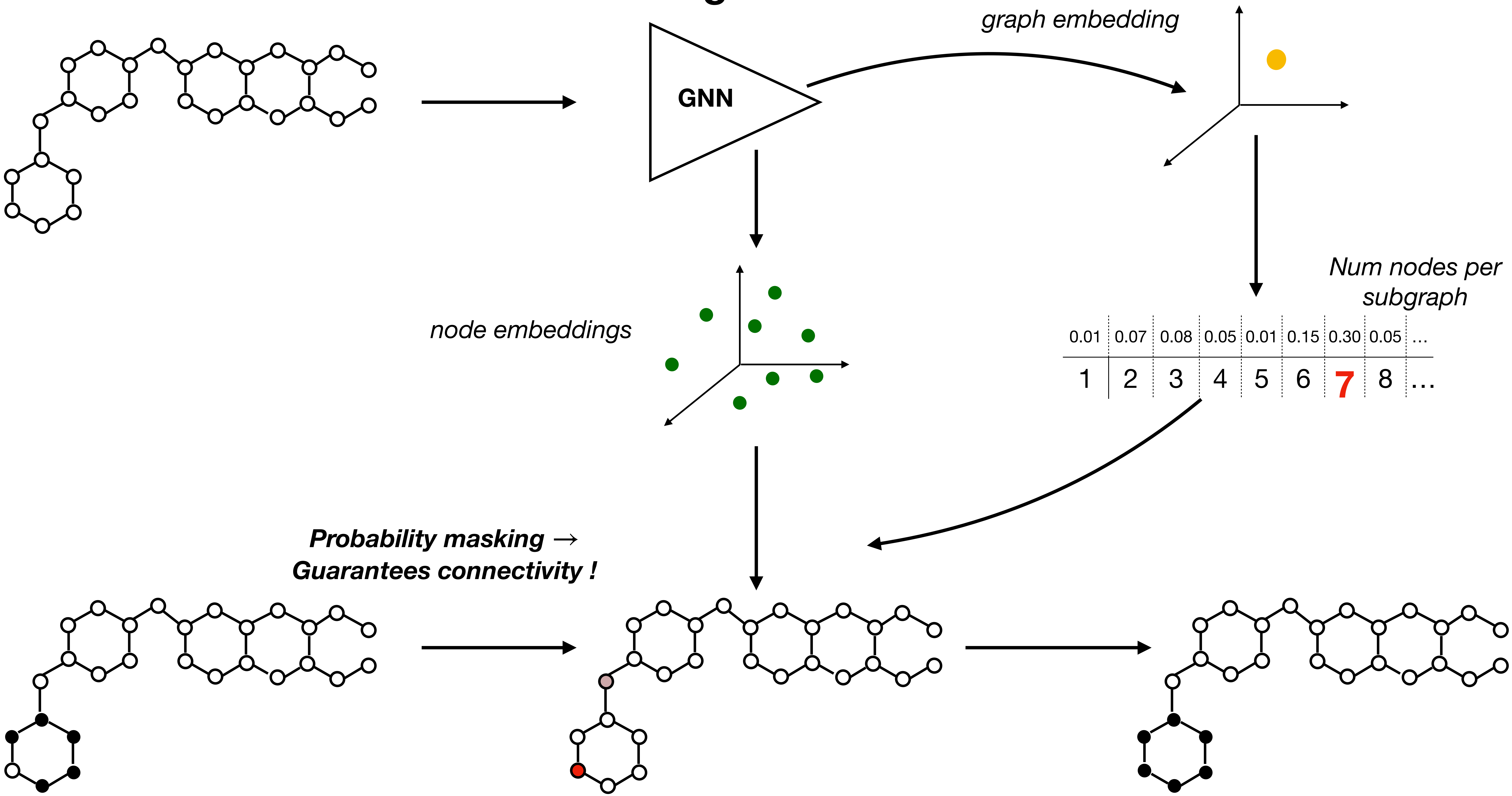
Learning to Partition



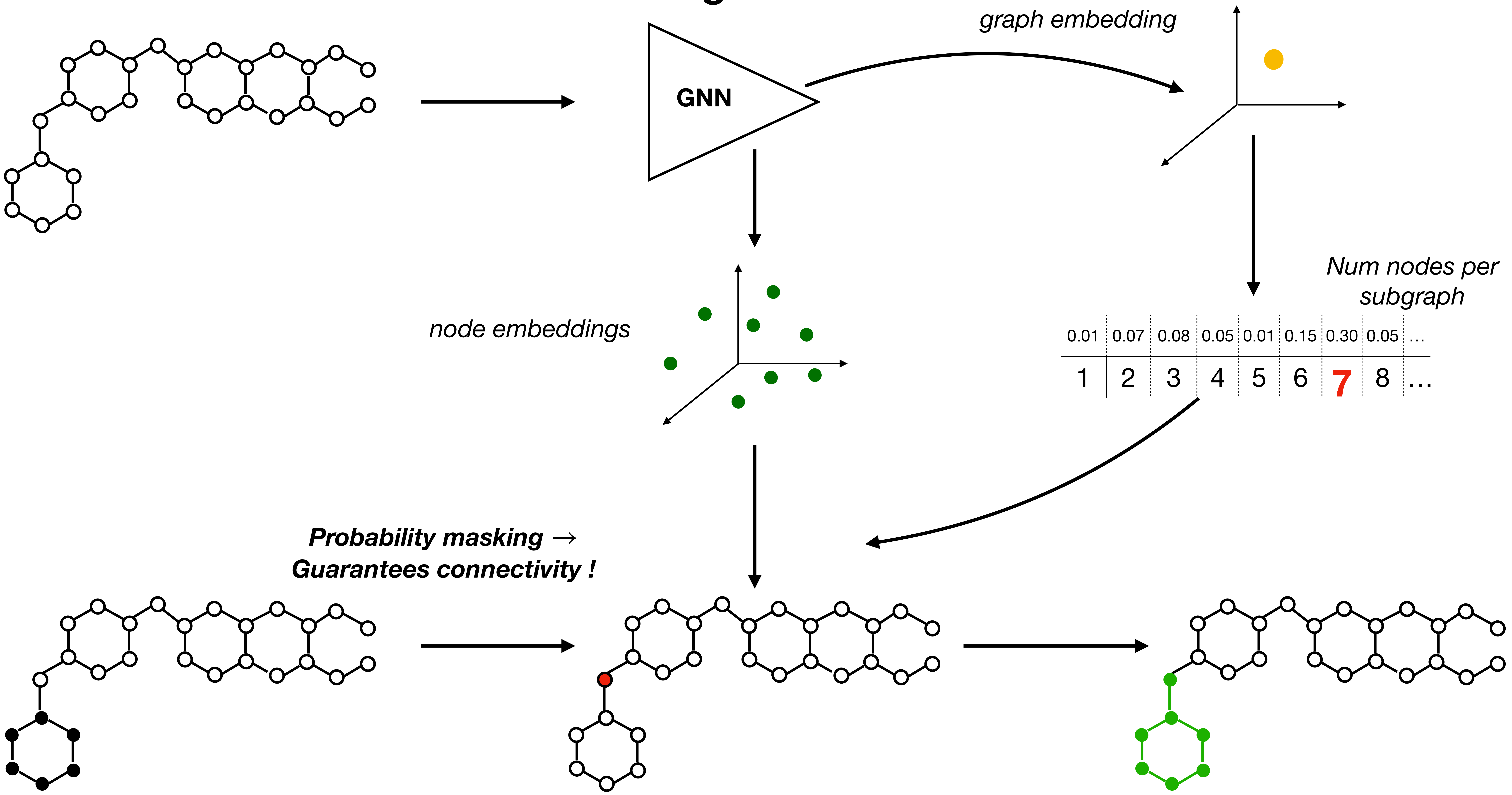
Learning to Partition



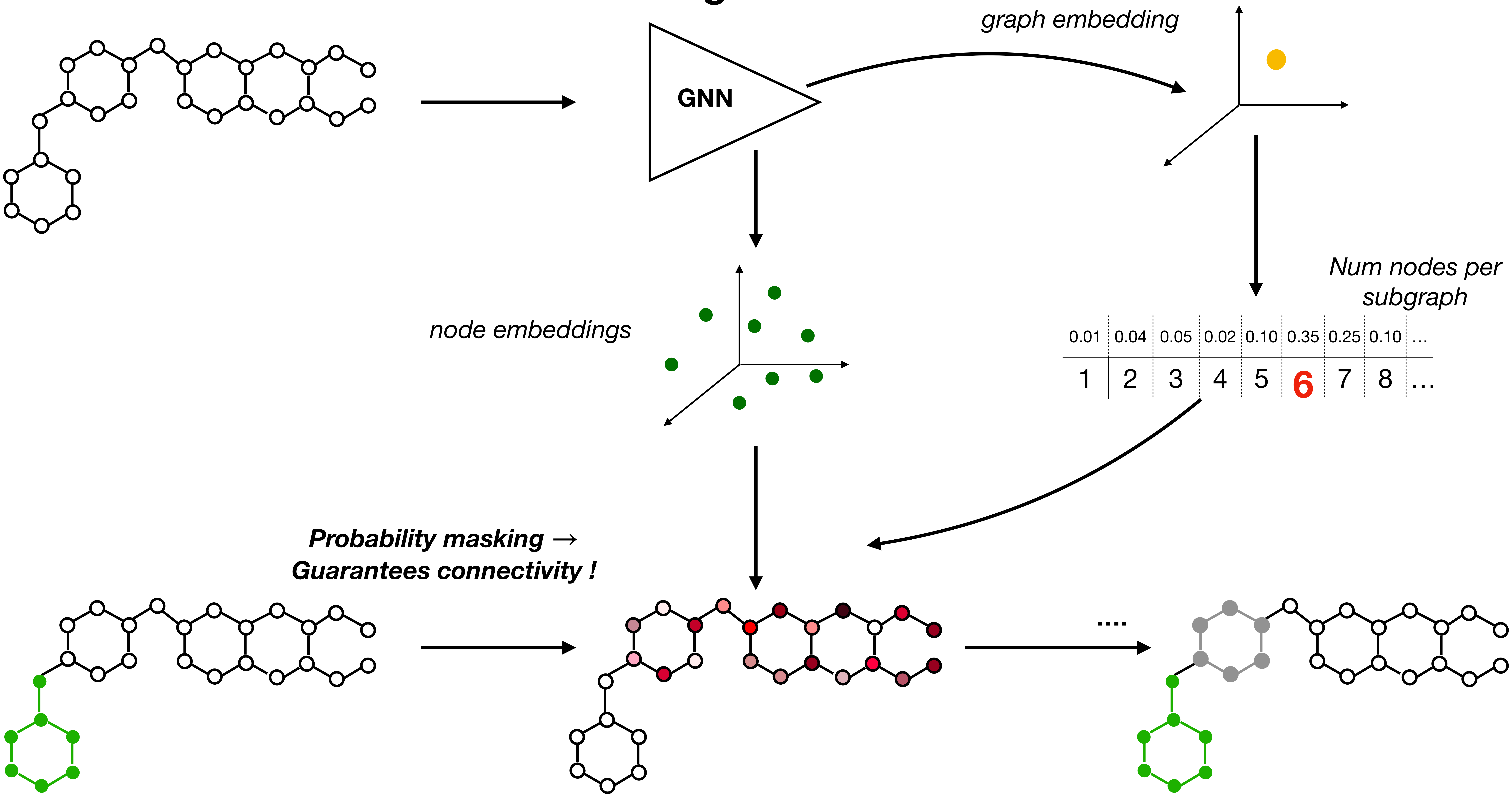
Learning to Partition



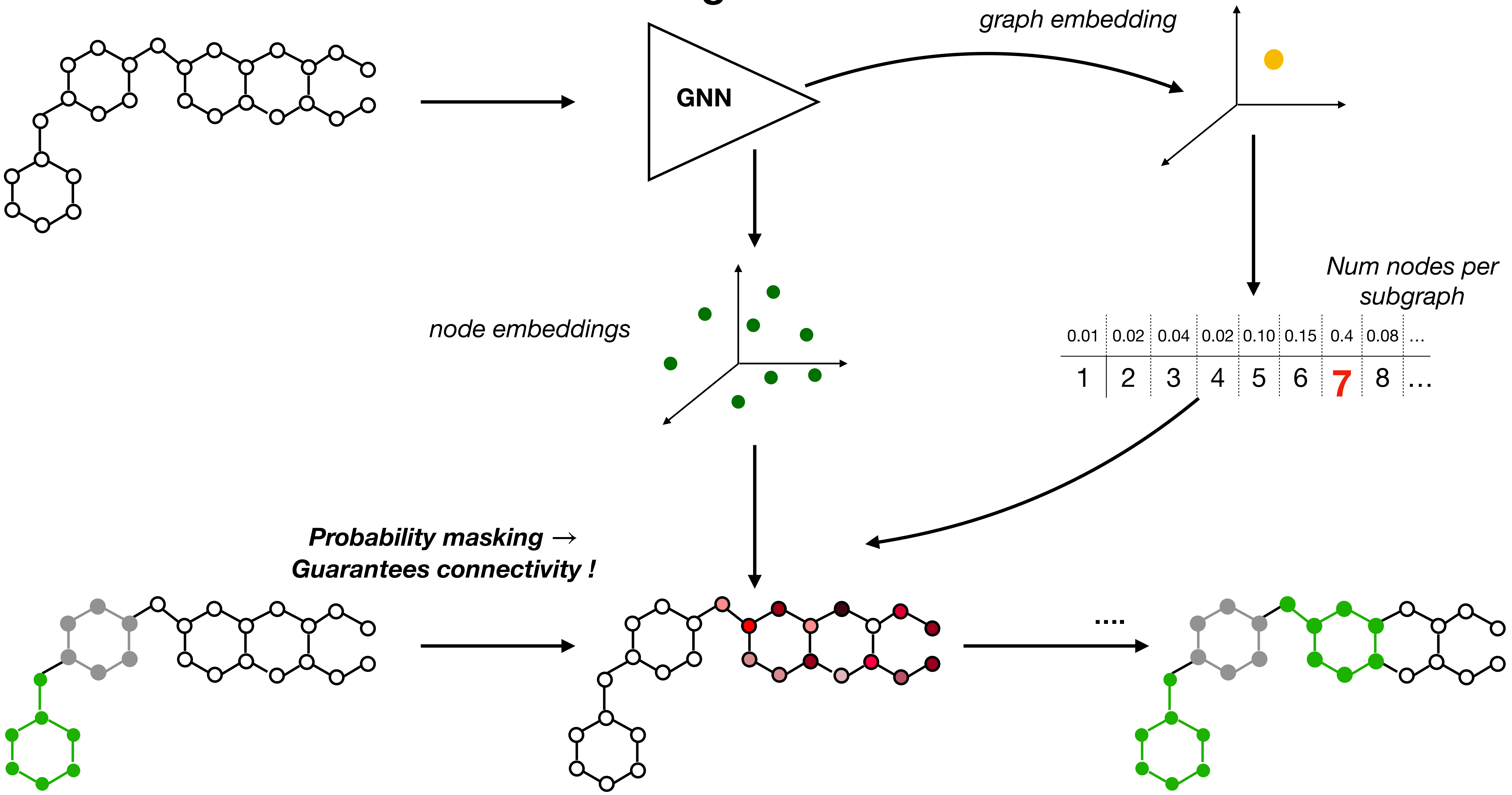
Learning to Partition



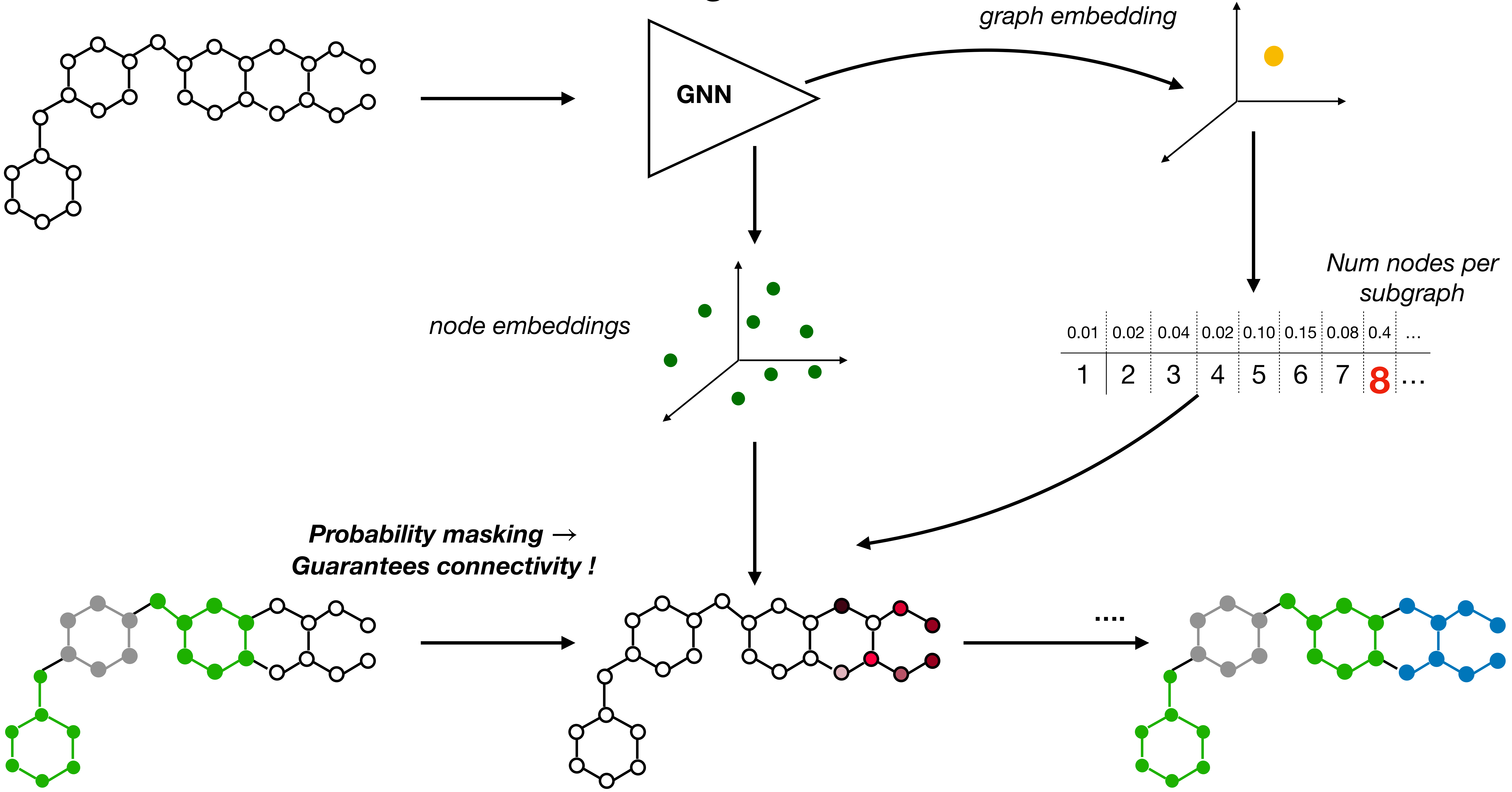
Learning to Partition



Learning to Partition



Learning to Partition



Optimisation

End-to-end with gradient descent:

(1) Differentiable w.r.t ϕ,

(2) D: continuous relaxation to obtain differentiability w.r.t. the fractional indicator variables $\hat{x} \in [0,1]$,

(3) PART$_\theta$ **:** REINFORCE.

Overall objective

$$\min_{\phi,\hat{\mathbf{x}},\theta} \sum_{G \in \mathcal{G}} \mathbb{E}_{p_\theta^{GNN}}[\mathrm{L}_{\phi,\hat{\mathbf{x}}}(\mathscr{H}, C \,|\, D)] + \mathrm{L}_{\hat{\mathbf{x}}}(D)$$

Theoretical gains

Quadratic *gains against (1) Null models and* ***Linear*** *against (2) Pure non-parametric partitioning.*

Theorem 1 (informal). *Consider a partitioning algorithm that decomposes a graph of n vertices into blocks of $k = O(1)$ vertices. Under mild conditions, it holds that:*

$$\mathbb{E}_{G\sim p}[L_{PnC}(G)] \lesssim \mathbb{E}_{G\sim p}[L_{Part}(G)] \lesssim \mathbb{E}_{G\sim p}[L_{null}(G)]$$

The absolute compression gains are:

$$\mathbb{E}_{G\sim p}[L_{part}(G)] \lesssim \mathbb{E}_{G\sim p}[L_{null}(G)] - \Theta(n^2) \quad \textit{and} \quad \mathbb{E}_{G\sim p}[L_{PnC}(G)] \lesssim \mathbb{E}_{G\sim p}[L_{Part}(G)] - \Theta(n)$$

Theoretical gains

Quadratic *gains against (1) Null models and* ***Linear*** *against (2) Pure non-parametric partitioning.*

Theorem 1 (informal). *Consider a partitioning algorithm that decomposes a graph of n vertices into blocks of $k = O(1)$ vertices. Under mild conditions, it holds that:*

$$\mathbb{E}_{G\sim p}[L_{PnC}(G)] \lesssim \mathbb{E}_{G\sim p}[L_{Part}(G)] \lesssim \mathbb{E}_{G\sim p}[L_{null}(G)]$$

The absolute compression gains are:

$$\mathbb{E}_{G\sim p}[L_{part}(G)] \lesssim \mathbb{E}_{G\sim p}[L_{null}(G)] - \Theta(n^2) \quad \textit{and} \quad \mathbb{E}_{G\sim p}[L_{PnC}(G)] \lesssim \mathbb{E}_{G\sim p}[L_{Part}(G)] - \Theta(n)$$

Linear *gains against (3) PnC without isomorphism testing.*

Theorem 2 (informal). *Consider a PnC compressor that yields dictionary subgraphs with probability $1-\delta$. Then:*

$$\mathbb{E}_{G\sim p}[\mathrm{L}_{PnC-S}(G)] \approx \mathbb{E}_{G\sim p}[\mathrm{L}_{PnC-G}(G)] - n(1-\delta)\log k$$

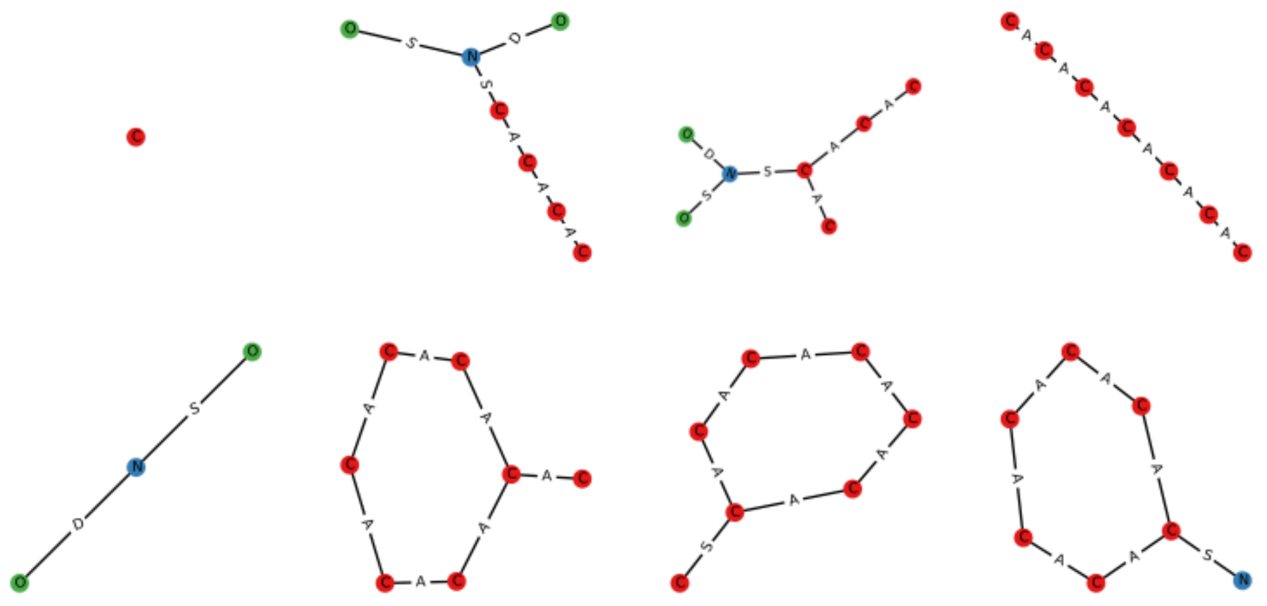
Results

- Biological and Social network distributions.
- Baselines:
  - (1) Null (uninformative),
  - (2) Pure partitioning,
  - (3) Deep generative models
- Description length is measured in **bits per edges.**

Results

Method type	Graph type	Small Molecules								
	Dataset name	MUTAG			PTC			ZINC		
		data	total	params	data	total	params	data	total	params
Null	Uniform (raw adjac.)	-	8.44	-	-	17.43	-	-	10.90	-
	Edge list	-	7.97	-	-	9.38	-	-	8.60	-
	Erdős-Renyi	-	4.78	-	-	5.67	-	-	5.15	-
Neural (likelihood)	GraphRNN	1.33	1669.77	388K	1.57	698.08	389K	1.62	22.39	388K
	GRAN	0.81	6279.28	1460K	2.18	2636	1470K	1.30	79.50	1461K
Partitioning (non-parametric)	SBM-Bayes	-	4.62	-	-	5.12	-	-	4.75	-
	Louvain	-	4.80	-	-	5.27	-	-	4.77	-
	PropClust	-	4.92	-	-	5.40	-	-	4.85	-
PnC	PnC + SBM	3.81	4.11	49	4.38	4.79	155	3.34	3.45	594
	PnC + Louvain	2.20	**2.51**	47	2.65	**3.15**	166	1.96	**1.99**	196
	PnC + PropClust	2.42	**3.03**	63	3.38	**4.02**	178	2.20	**2.35**	726
	PnC + Neural Part.	2.17±0.02	**2.45±0.02**	46±1	2.63±0.26	**2.97±0.14**	143±31	2.01±0.02	**2.07±0.03**	384±105

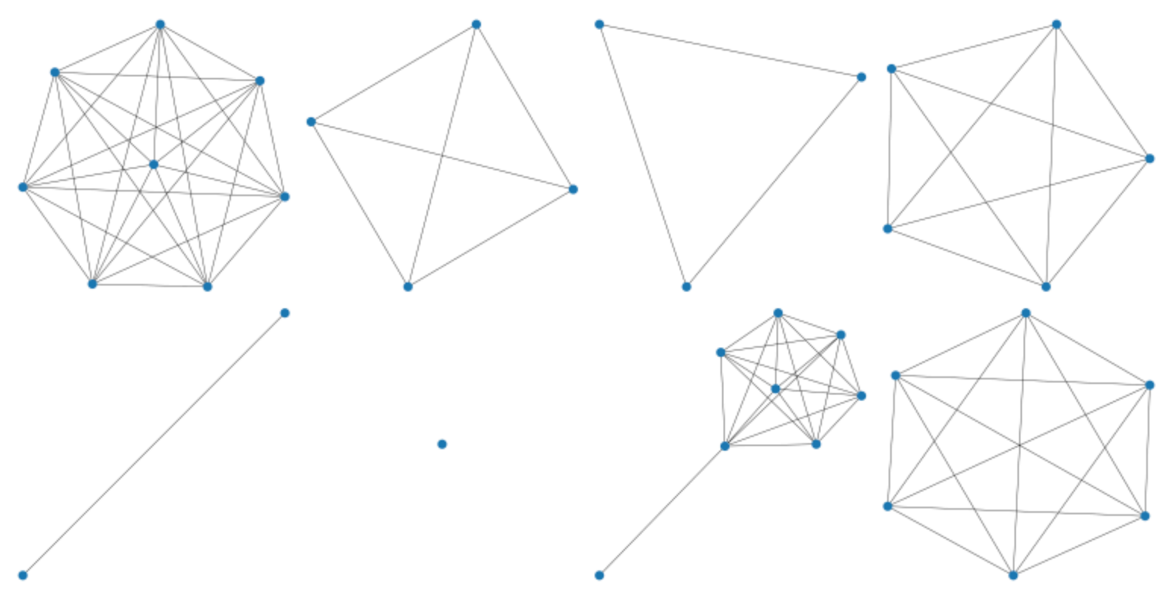
- **Deep generative models** suffer from overparametrisation.
- **Learning to partition** helps for graphs with recurrent substructures



Results

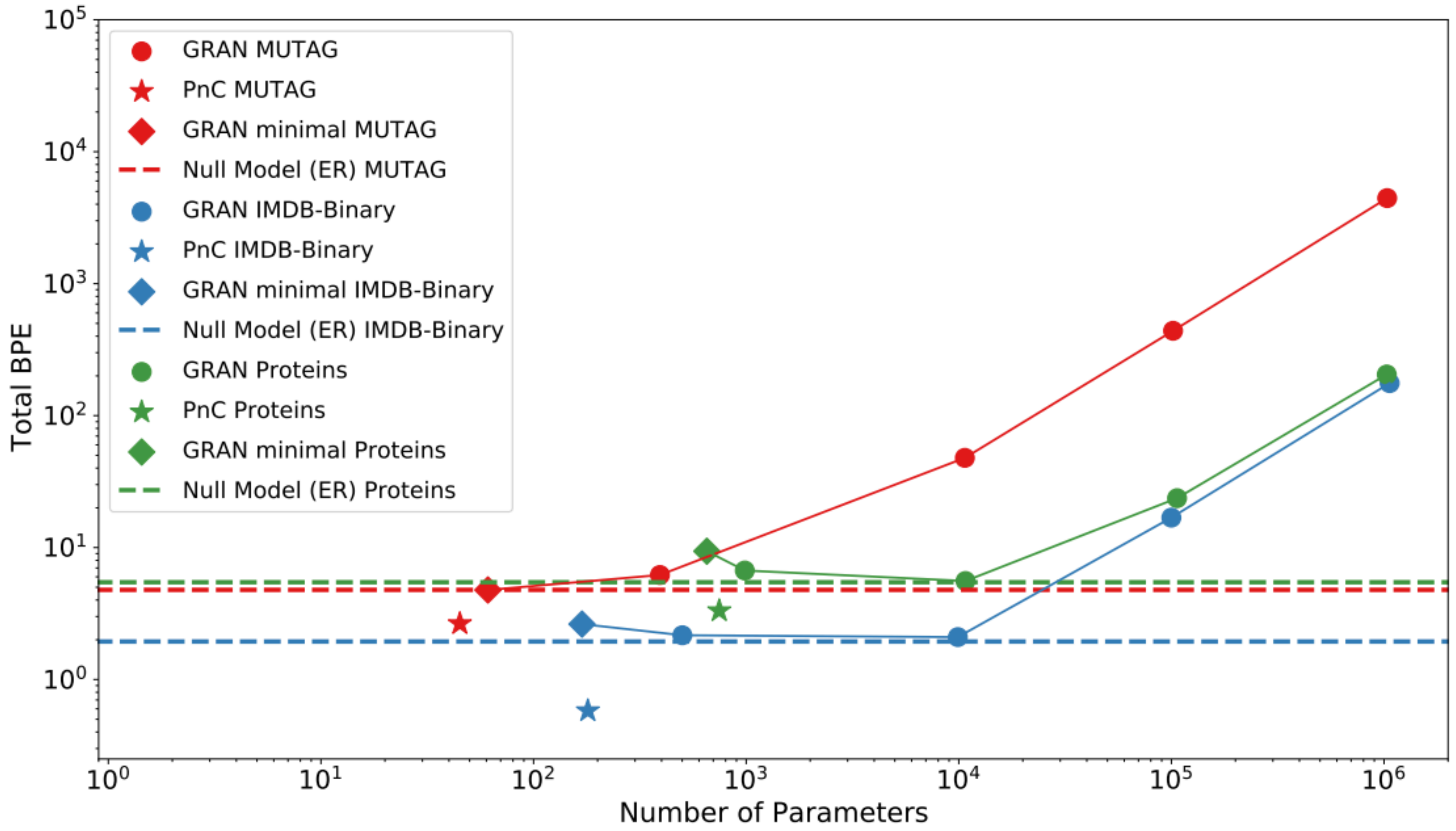
Method type	Graph type	Biology			Social Networks					
	Dataset name	PROTEINS			IMDB-B			IMDB-M		
		data	total	params	data	total	params	data	total	params
Null (non-parametric)	Uniform (raw adjac.)	-	24.71	-	-	2.52	-	-	1.83	-
	Edge list	-	10.92		-	8.29	-	-	7.74	-
	Erdős-Renyi	-	5.46	-	-	1.94	-	-	1.32	-
Neural (likelihood)	GraphRNN	2.03	79.51	392K	1.03	66.65	395K	0.72	64.28	392K
	GRAN	1.51	304.735	1545K	0.26	244.57	1473K	0.17	237.65	1467K
Partitioning (non-parametric)	SBM-Bayes	-	3.98	-	-	**0.80**	-	-	**0.60**	-
	Louvain	-	3.95	-	-	1.22	-	-	0.88	-
	PropClust	-	4.11	-	-	1.99	-	-	1.37	-
PnC	PnC + SBM	3.26	3.60	896	0.50	**0.54**	198	0.38	**0.38**	157
	PnC + Louvain	3.30	**3.58**	854	0.96	**1.02**	202	0.66	**0.70**	141
	PnC + PropClust	3.40	**3.70**	866	1.45	1.64	241	0.93	1.04	178
	PnC + Neural Part.	3.34±0.25	**3.51±0.23**	717±61	1.00±0.04	1.05±0.04	186±25	0.66±0.05	0.72±0.05	178±14

- **Partitioning-based** are strong baselines for graphs with community structure.
- **PnC** consistently improves compression in every case.



PnC vs Overparametrised NNs

dataset	**GraphRNN**	**GRAN**
MUTAG	x1264	x3412
PTC	x484	x3173
ZINC	x38	x90
PROTEINS	x60	x168
IMDBB	infeasible	x763
IMDBM	infeasible	x1033

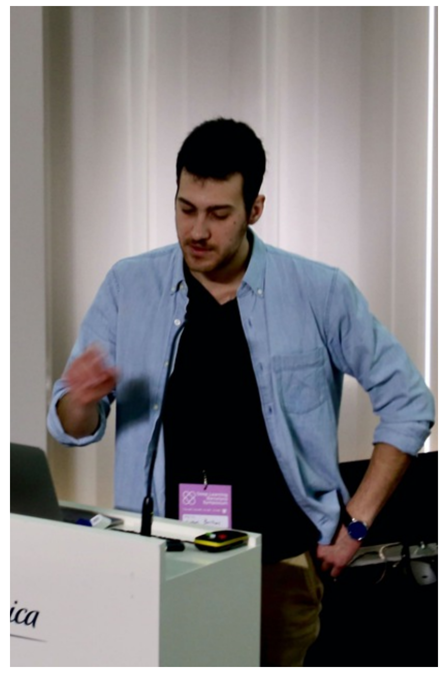


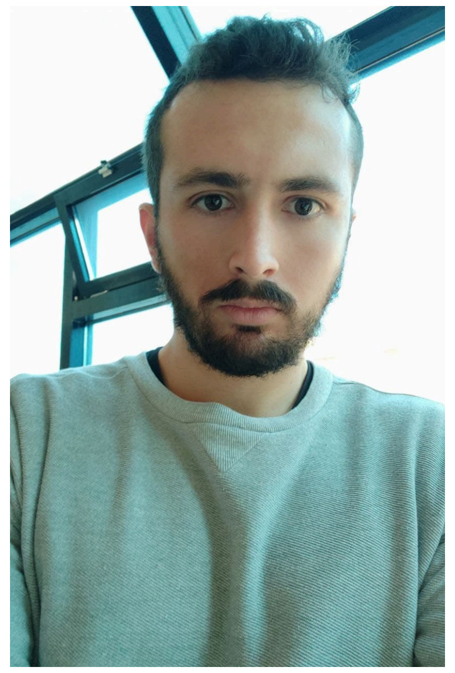
- Vanilla graph generators are suboptimal for compression **(heavily overparametrised!).**
- Unclear how to minimise total description length.
- Posthoc model compression: tedious/often ineffective.

Take home messages

- Lossless graph compression requires estimating distributions over isomorphism classes.
- Challenging in various respects (*computationally*, *statistically*, *expressivity*).
- PnC provides desirable tradeoffs w.r.t the above with guaranteed compression gains.
- Additional advantages: *explainability* + also optimising w.r.t. the *model description length.*
- Can we do better?
    1. Learnable Partitioning - problem at its own sake.
    2. How to scale to large graphs?
    3. Deep generative models + accounting for the total DL during training *(general problem in neural compression).*

Interested to know more? Let’s chat in the NeurIPS poster session!






@gbouritsas

arxiv.org/abs/2107.01952