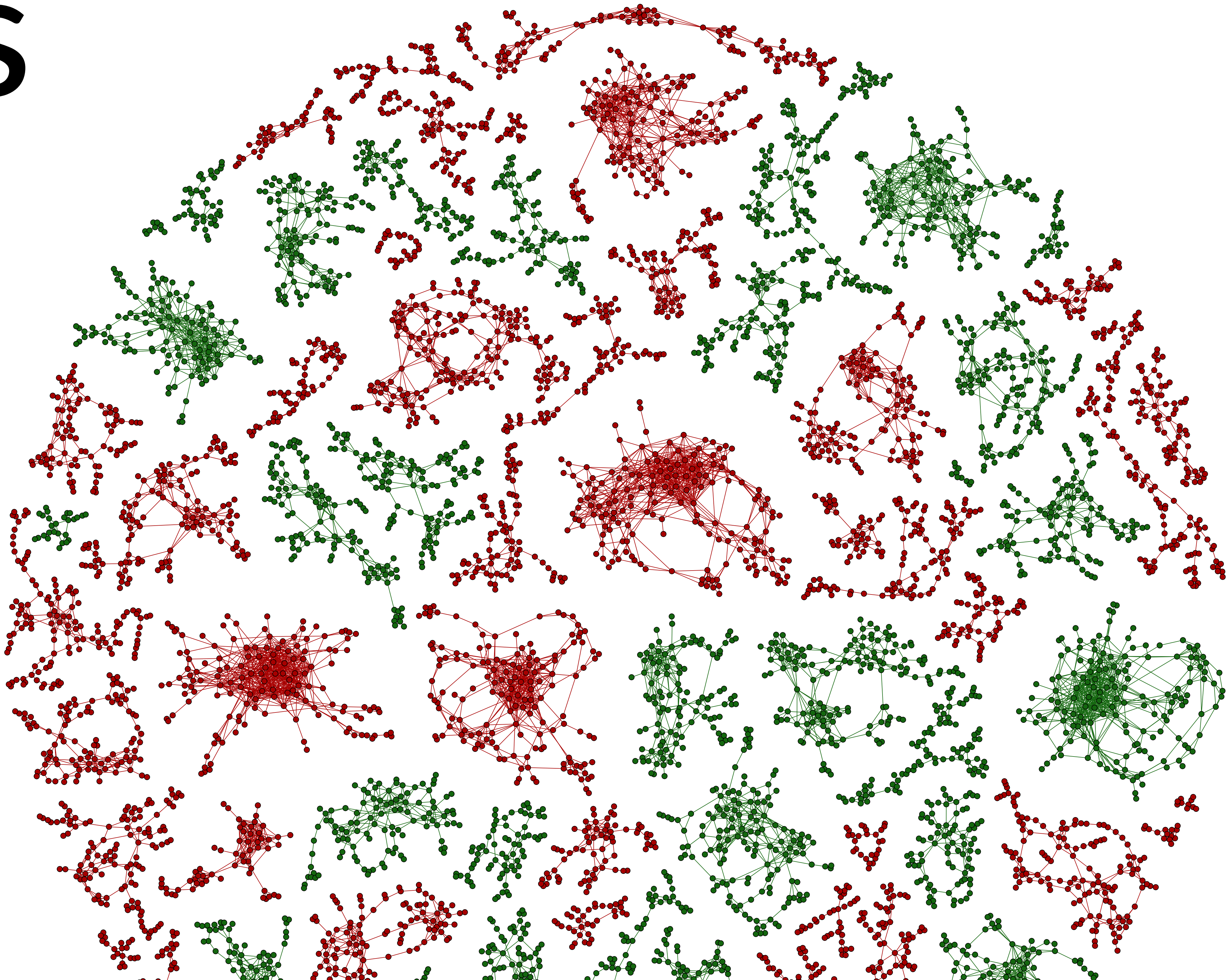
Similarities and Representations of Graph Structures

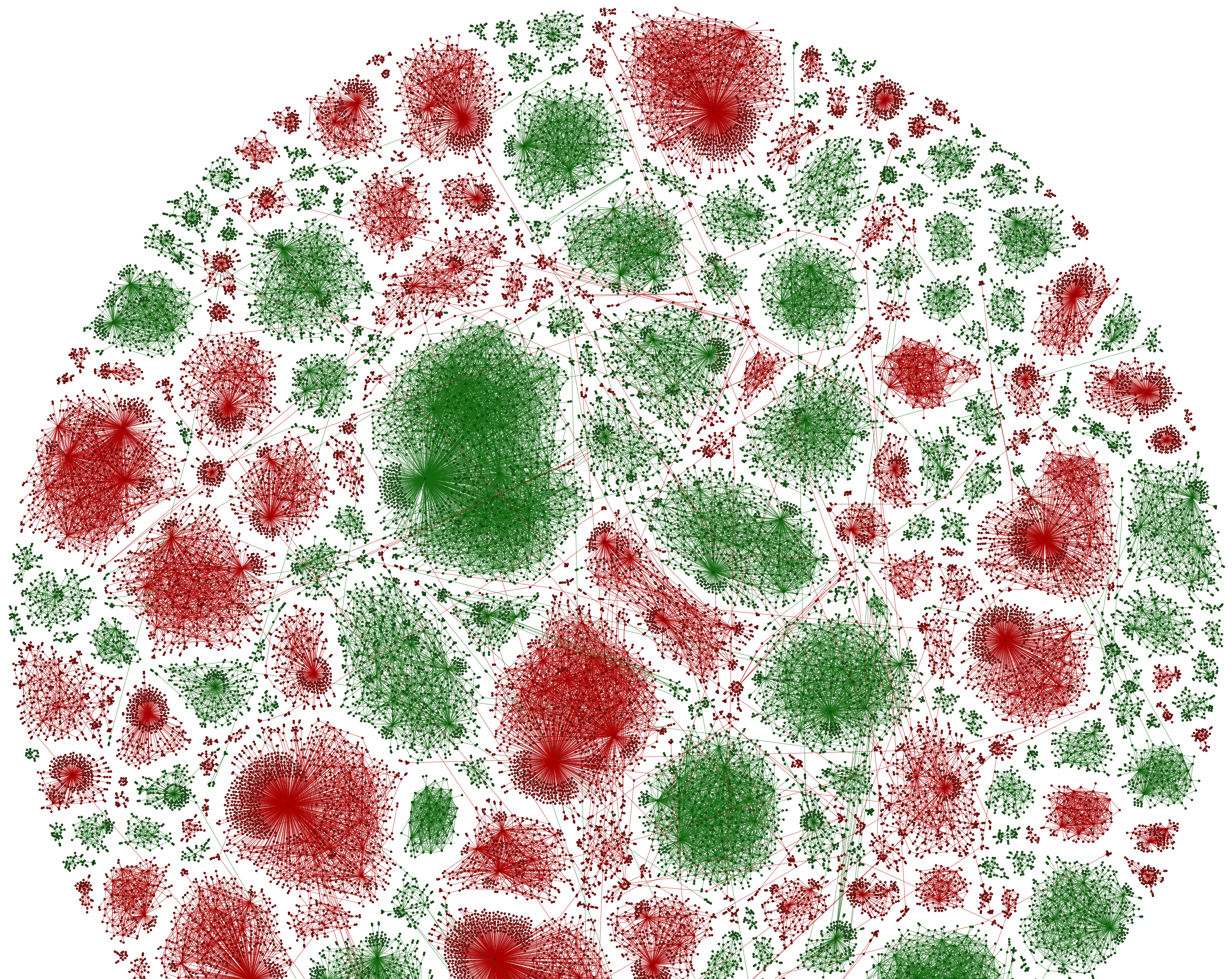
Anton Tsitsulin

UNIVERSITÄT BONN



Introduction

On the importance of **graphs, similarities,** and **representations**

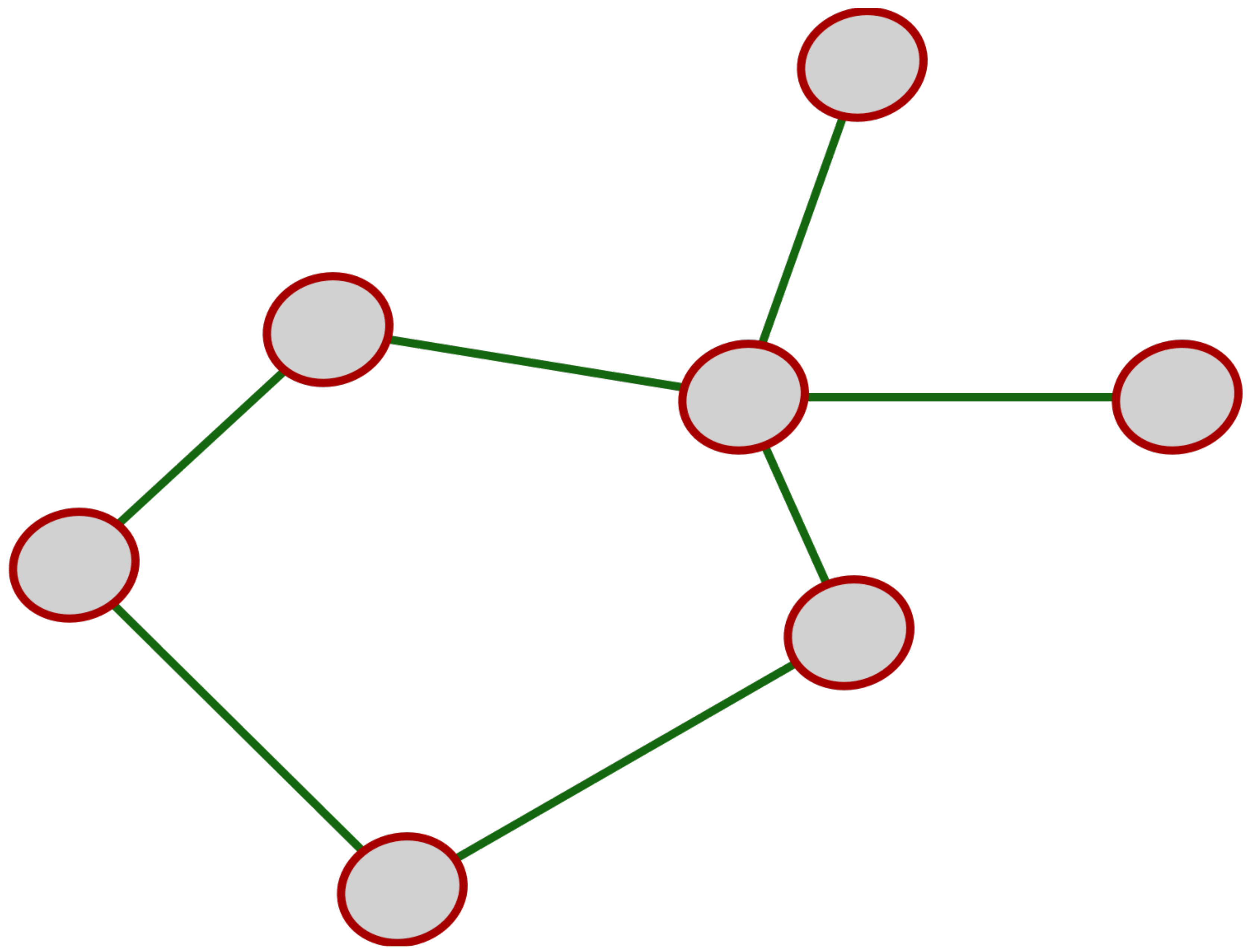


What are graphs?

Graphs are **representations** of relationships (**edges**) between entities (**nodes**).

There are two important types:

1. Natural graphs are graphs in which edges naturally come from an external source.
2. Similarity graphs in which the edge relationship is based on some **similarity measure** between nodes.



Examples of natural graphs:
social networks, protein interactomes, road graphs, co-purchase graphs, …

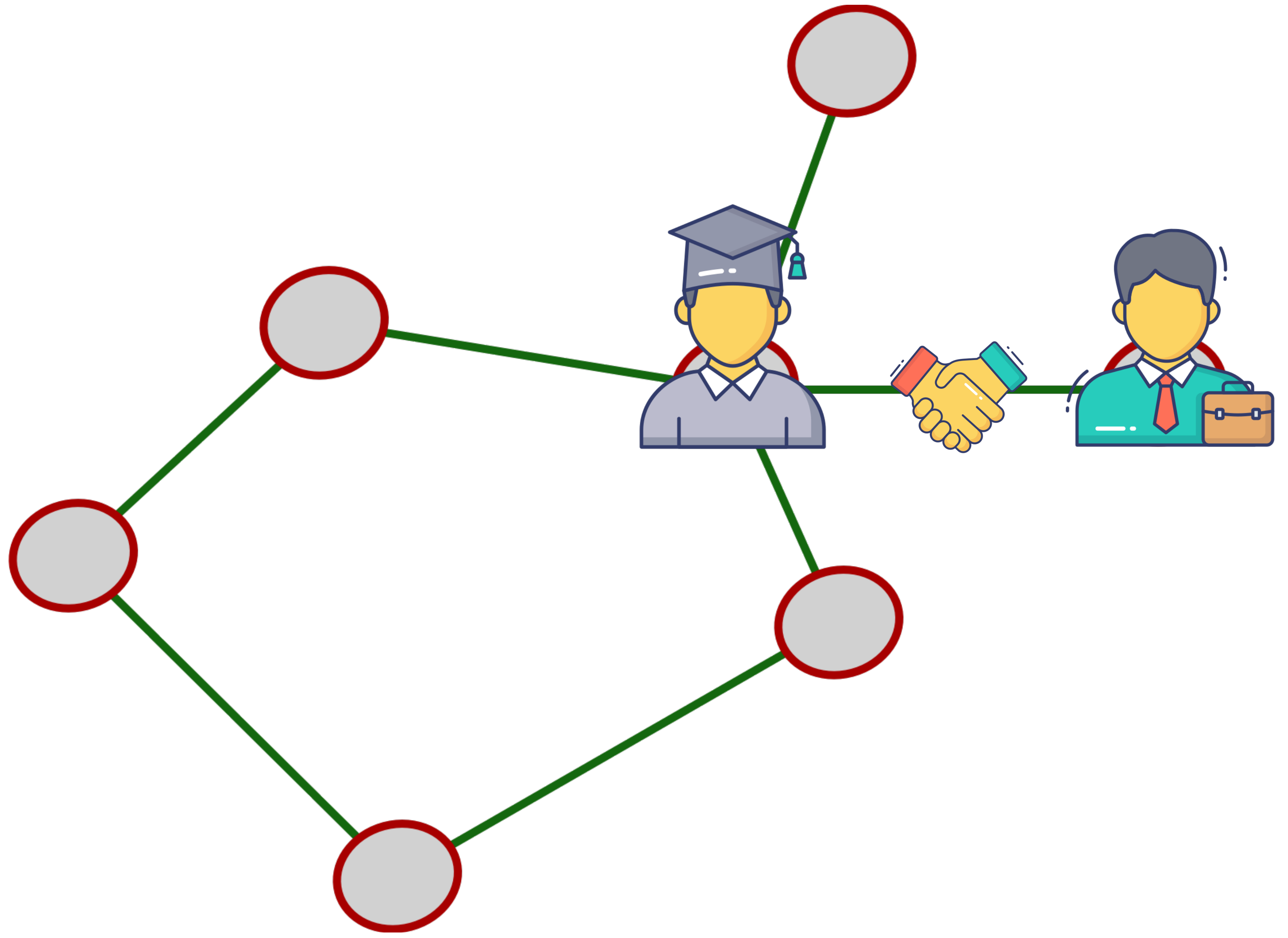
What are graphs?

Graphs are **representations** of relationships (**edges**) between entities (**nodes**).

There are two important types:

1. Natural graphs are graphs in which edges naturally come from an external source.

2. Similarity graphs in which the edge relationship is based on some **similarity measure** between nodes.



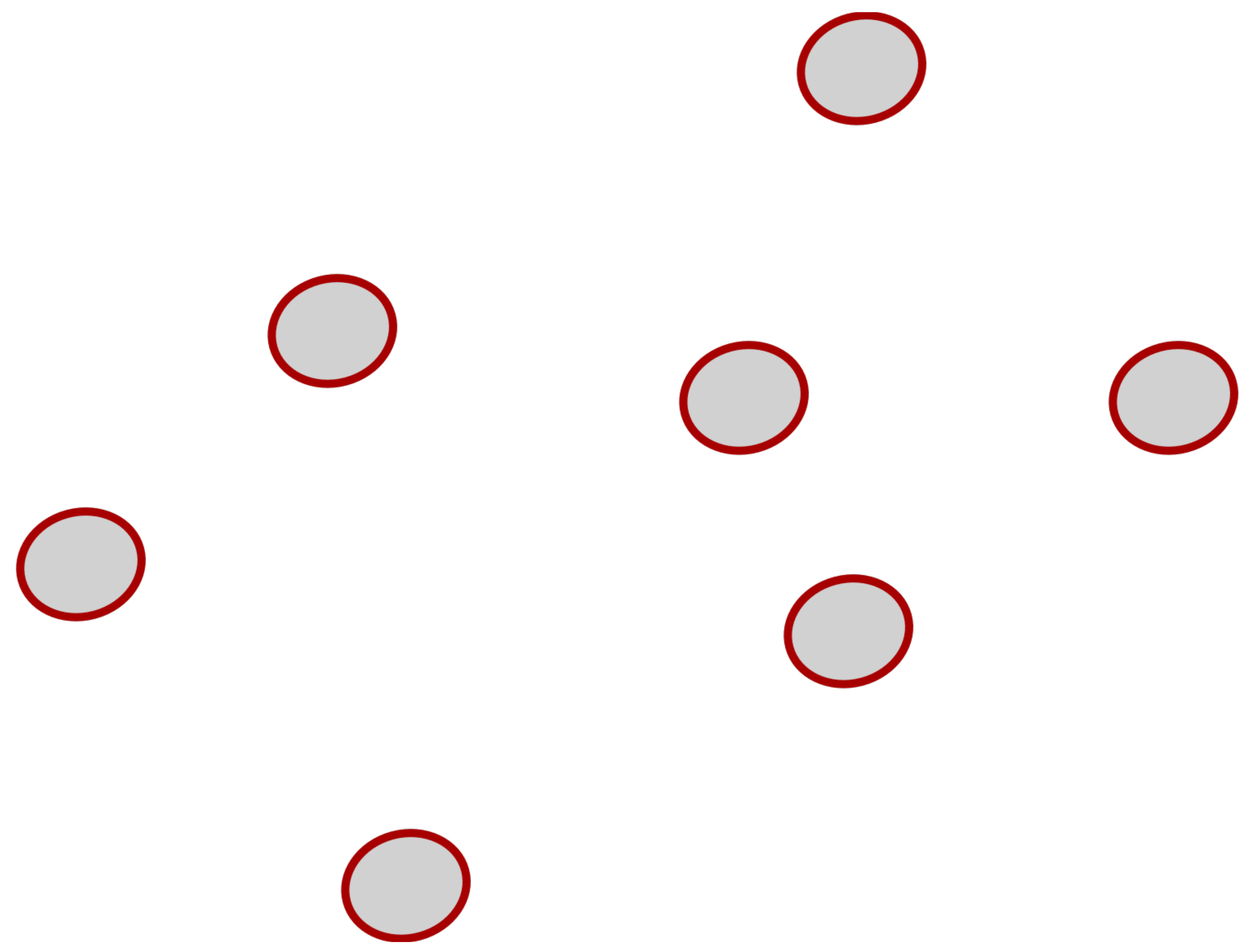
Examples of natural graphs:
social networks, protein interactomes, road graphs, co-purchase graphs, …

What are graphs?

Graphs are **representations** of relationships (**edges**) between entities (**nodes**).

There are two important types:

1. Natural graphs are graphs in which edges naturally come from an external source.
2. Similarity graphs in which the edge relationship is based on some **similarity measure** between nodes.



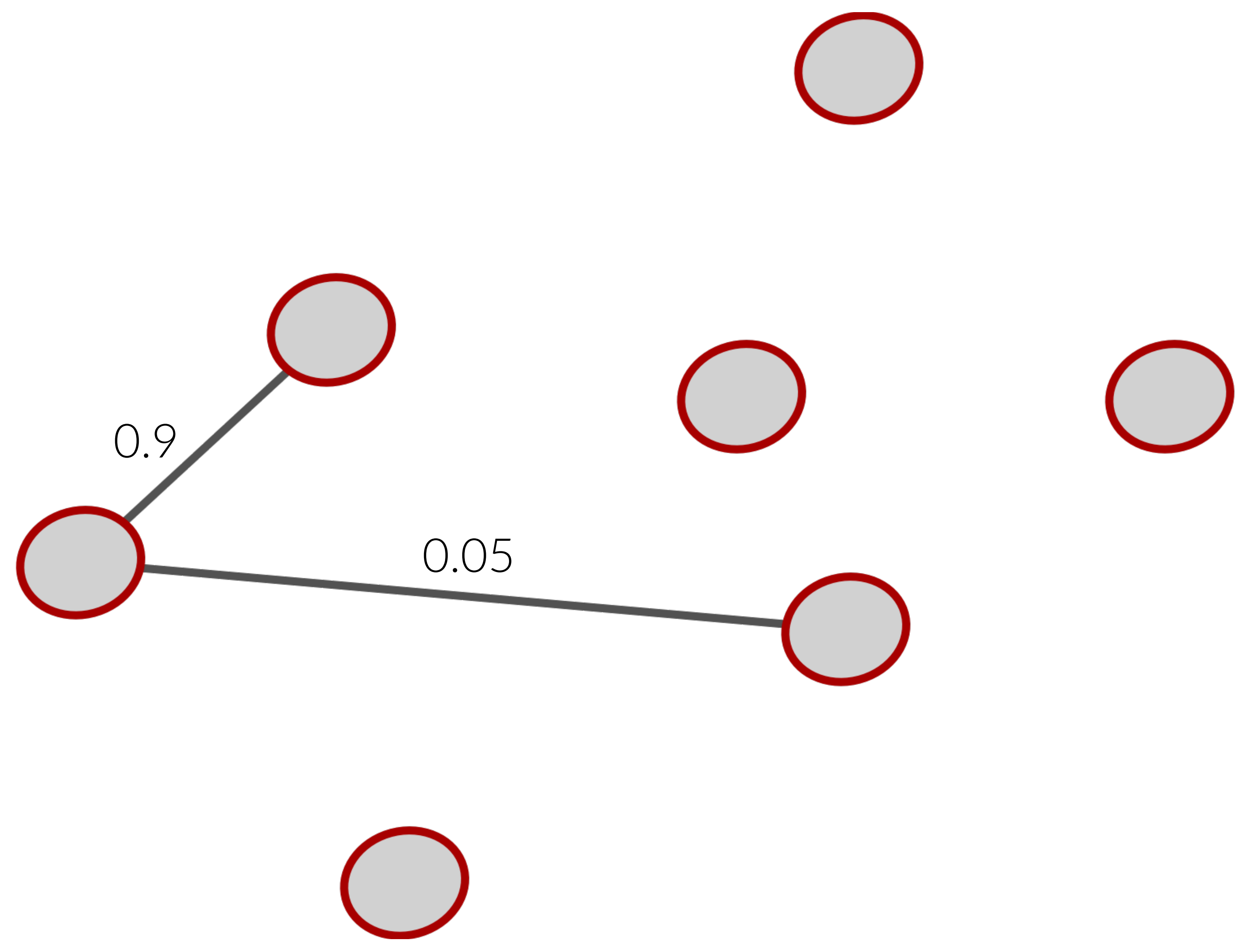
We can build similarity graphs from any type of data!

What are graphs?

Graphs are **representations** of relationships (**edges**) between entities (**nodes**).

There are two important types:

1. Natural graphs are graphs in which edges naturally come from an external source.
2. Similarity graphs in which the edge relationship is based on some **similarity measure** between nodes.



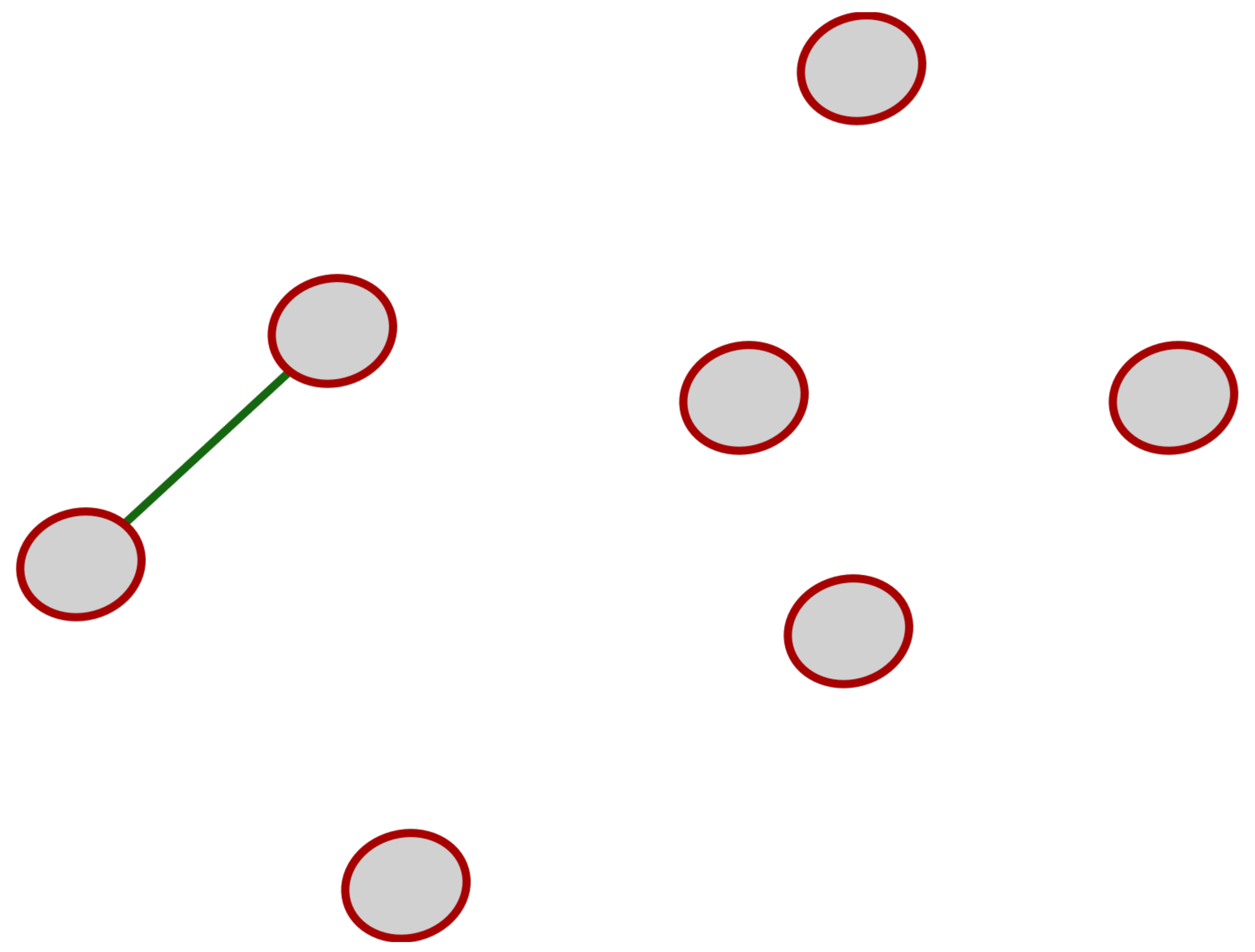
We can build similarity graphs from any type of data!

What are graphs?

Graphs are **representations** of relationships (**edges**) between entities (**nodes**).

There are two important types:

1. Natural graphs are graphs in which edges naturally come from an external source.
2. Similarity graphs in which the edge relationship is based on some **similarity measure** between nodes.



We can build similarity graphs from any type of data!

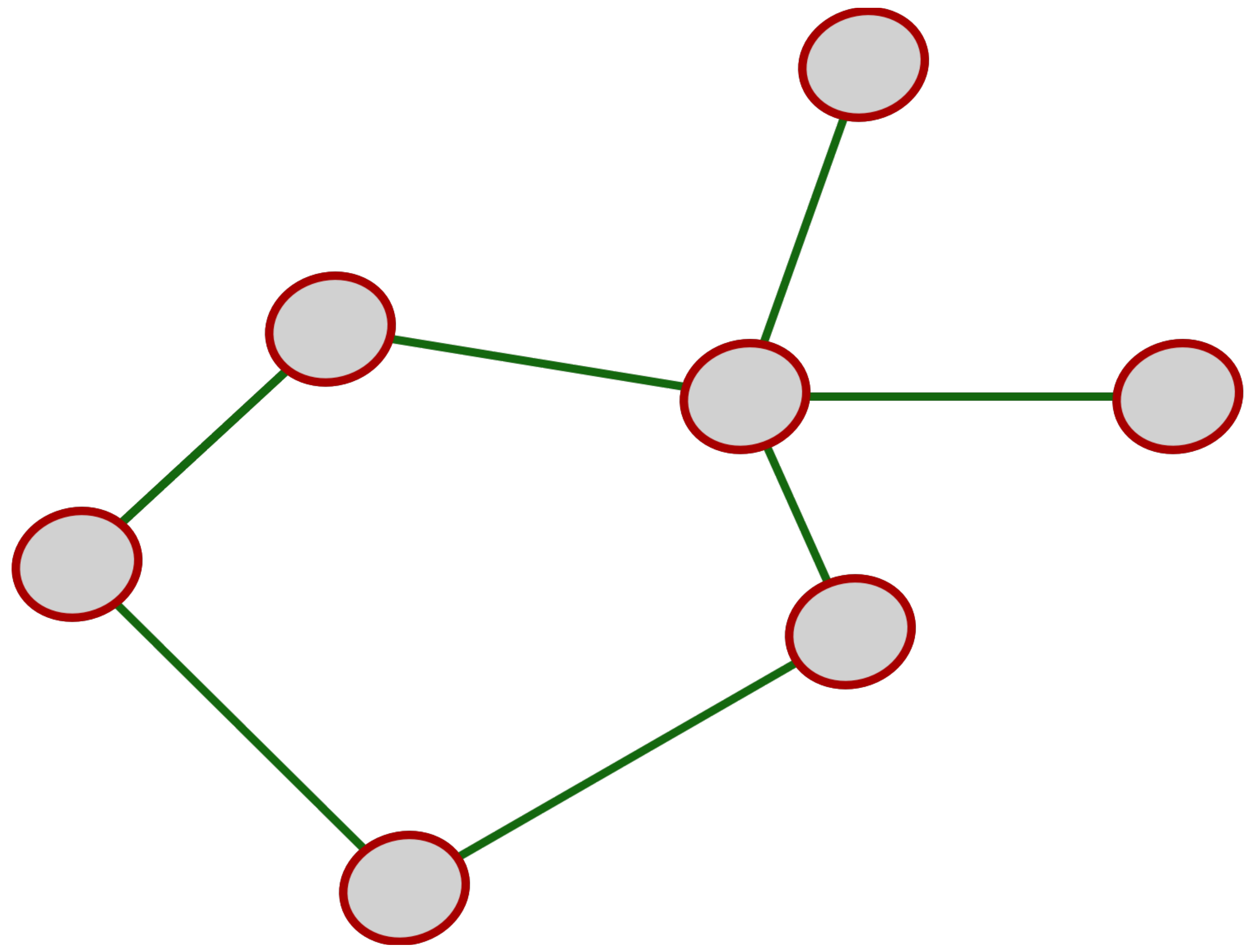
What are graphs?

Graphs are **representations** of relationships (**edges**) between entities (**nodes**).

There are two important types:

1. Natural graphs are graphs in which edges naturally come from an external source.

2. Similarity graphs in which the edge relationship is based on some **similarity measure** between nodes.



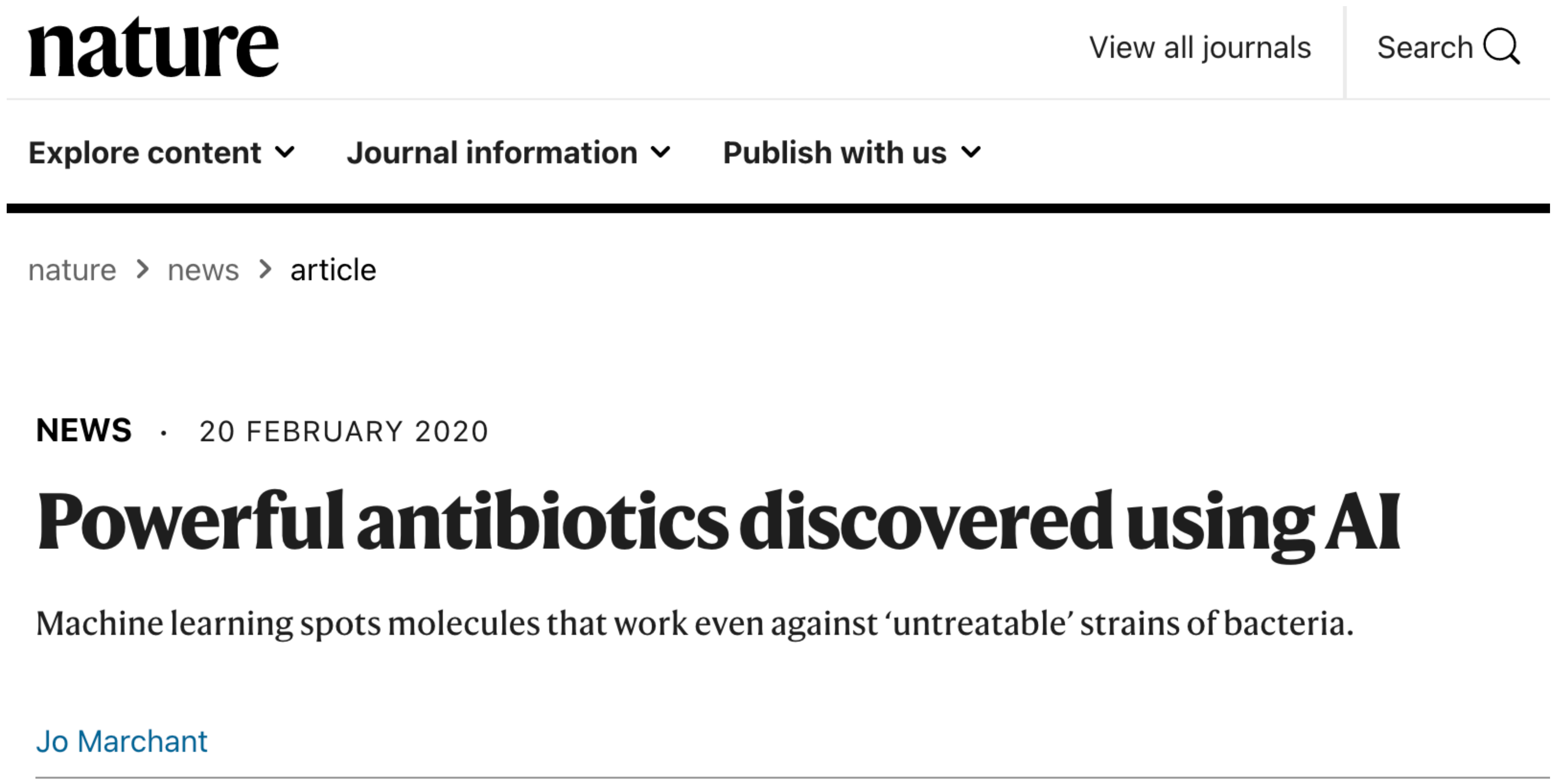
We can build similarity graphs from any type of data!

Graphs are everywhere

Graphs have numerous applications in biology, social sciences, physics and machine learning.

Typically, we want predictions on the nodes or whole graphs.

Graphs in the real world are large and numerous.



nature

View all journals | Search

Explore content | Journal information | Publish with us

nature > news > article

NEWS · 20 FEBRUARY 2020

Powerful antibiotics discovered using AI

Machine learning spots molecules that work even against ‘untreatable’ strains of bacteria.

Jo Marchant

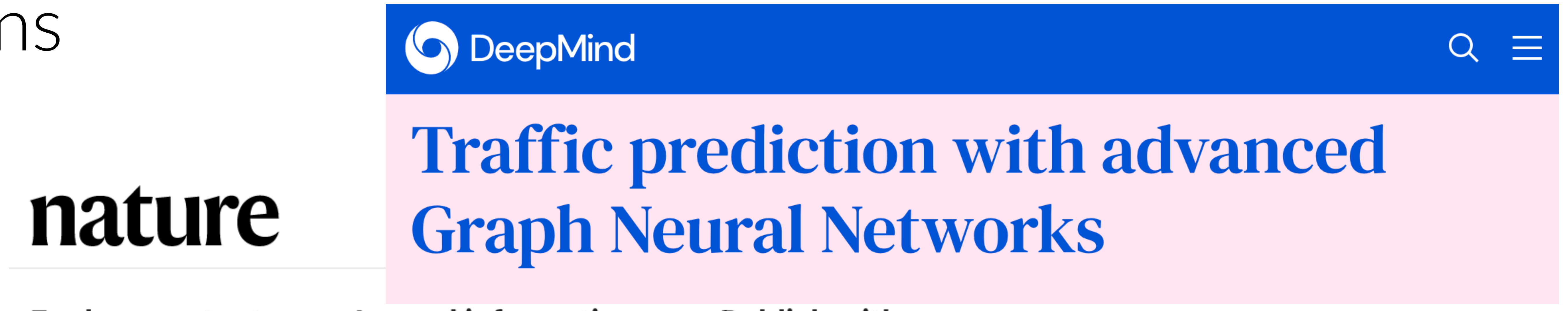
(Stokes *et al.*, Cell’20)

Graphs are everywhere

Graphs have numerous applications in biology, social sciences, physics and machine learning.

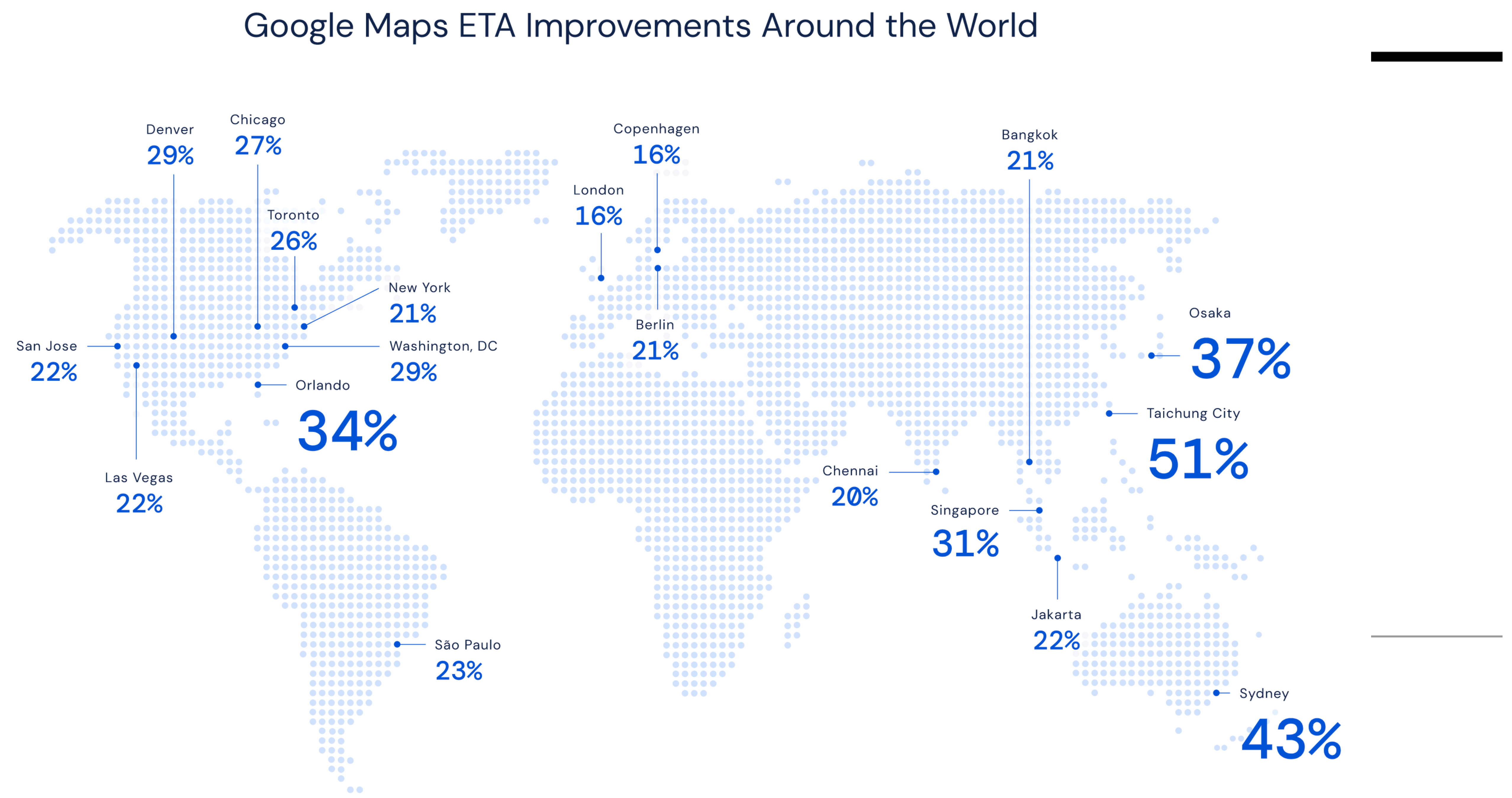
Typically, we want predictions on the nodes or whole graphs.

Graphs in the real world are large and numerous.


nature

DeepMind

Traffic prediction with advanced Graph Neural Networks



Similarities

A notion of similarity* between objects is key for data analysis.

If we have such notion, it is easy to do:

1. Classification with kernel-based methods
2. Clustering with linkage algorithms
3. Anomaly detection with distance heuristics
4. A whole lot of other tasks!

* a pseudometric

Representations

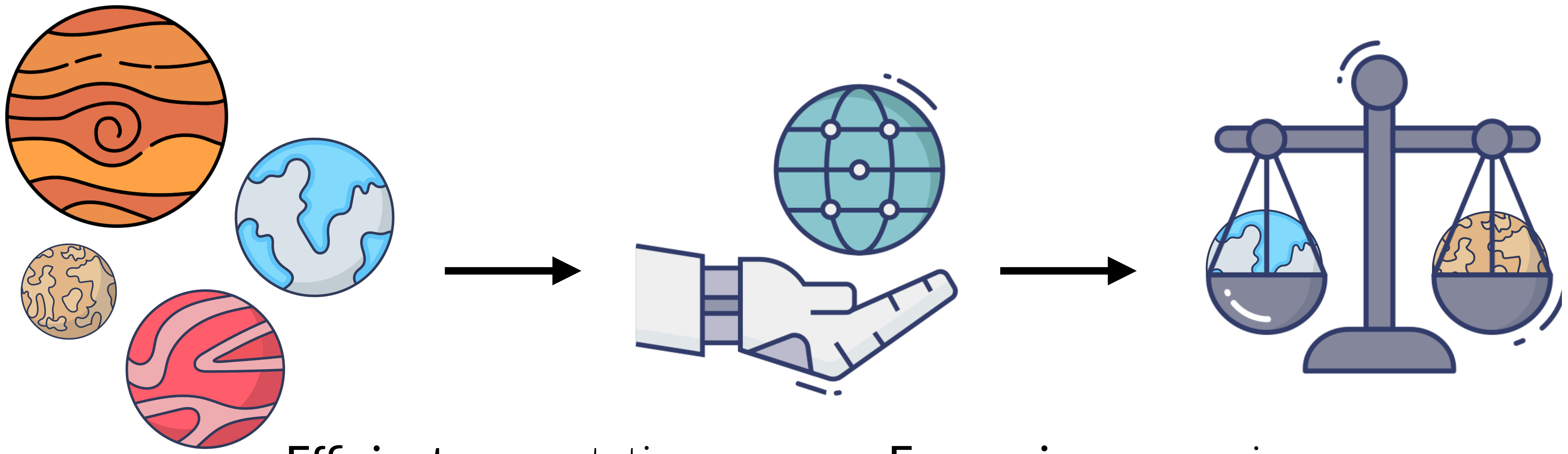
Suitable representation is crucial for efficient similarity computation.

Representations that lie in some low-dimensional space are called **explicit**.
They are more favourable than those with implicit comparisons:

1. Faster similarity computation
2. Approximate nearest neighbour search
3. Easier indexing & advanced range queries

The world is built for dense Euclidean data!

Representation and similarity design

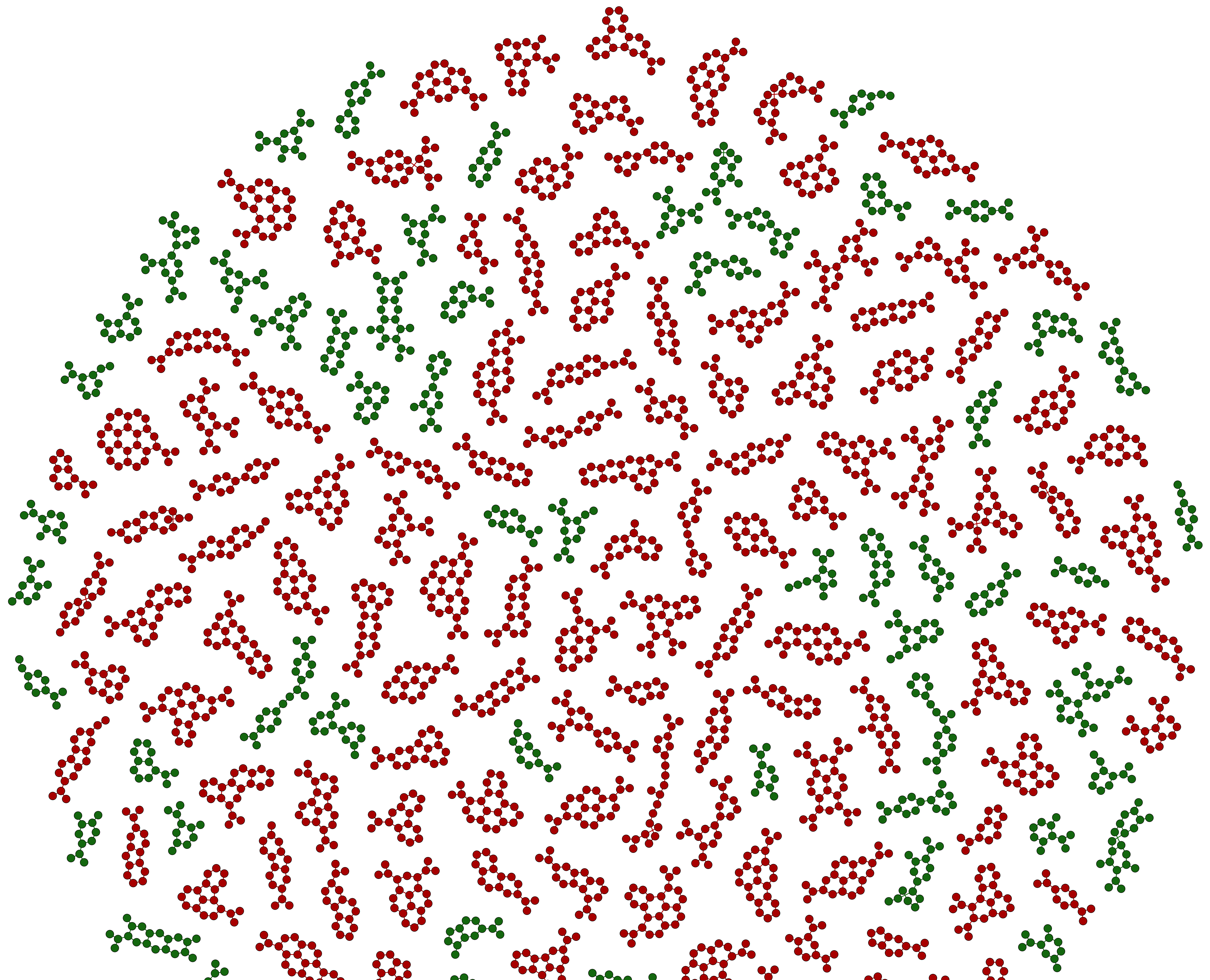


Efficient computation
of representations

Expressive comparison
of objects

Thesis overview

Core **proposal,** chapter **summary,** and main **contributions**



Core thesis proposal

Scalable, theoretically grounded algorithms
for building **explicit, expressive** representations
of graph-structured data

Core thesis proposal

Because graphs in the real world are big and numerous

Scalable, theoretically grounded algorithms for building **explicit, expressive** representations of graph-structured data

Core thesis proposal

For explainability and to ensure the performance of the algorithms

Scalable, theoretically grounded algorithms
for building **explicit, expressive** representations
of graph-structured data

Core thesis proposal

Scalable, theoretically grounded algorithms
for building **explicit, expressive** representations
of graph-structured data

Because the world is built
for dense Euclidean data

Core thesis proposal

Scalable, theoretically grounded algorithms
for building **explicit, expressive** representations
of graph-structured data

With excellent performance
across application domains

Thesis structure

Representations of nodes

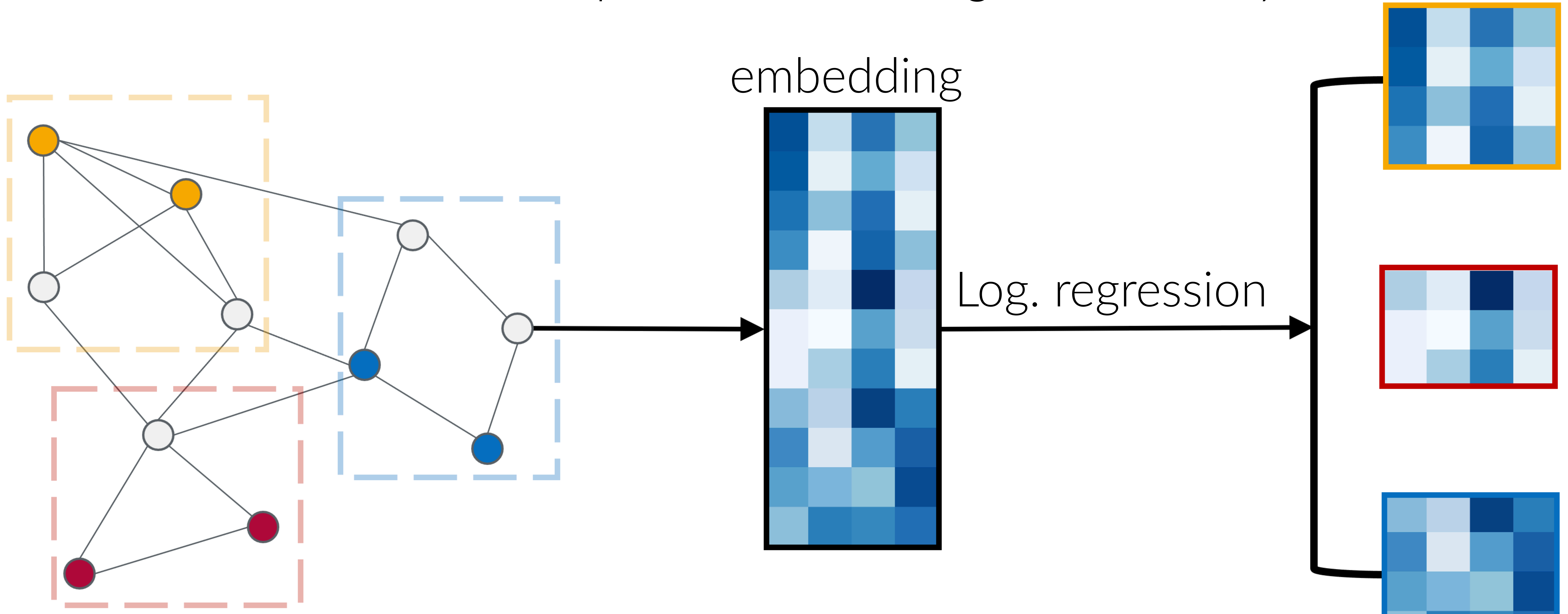
- Chapter 4: Versatile representations from similarity measures [**T**MKM, WWW'18]
- Chapter 5: Anytime node embeddings with quality guarantees [**T**MMKOM, VLDB'21]
- Chapter 6: Local node embeddings via approximate node similarity [P**T**ATSB, in subm.]
- Chapter 7: Node representations for clustering [**T**PPM, in submission]

Representations of graphs

- Chapter 9: Spectral optimal transport-based graph representations [**T**MKBM, KDD'18]
- Chapter 10: Self-supervised spectral graph representations [**T**MKBM]
- Chapter 11: Scalable approximations for graph embeddings [**T**MB, WWW'20]
- Chapter 12: ML applications: comparing data distributions [**T**MMKBOM, ICLR'20]

Introduction to node representations

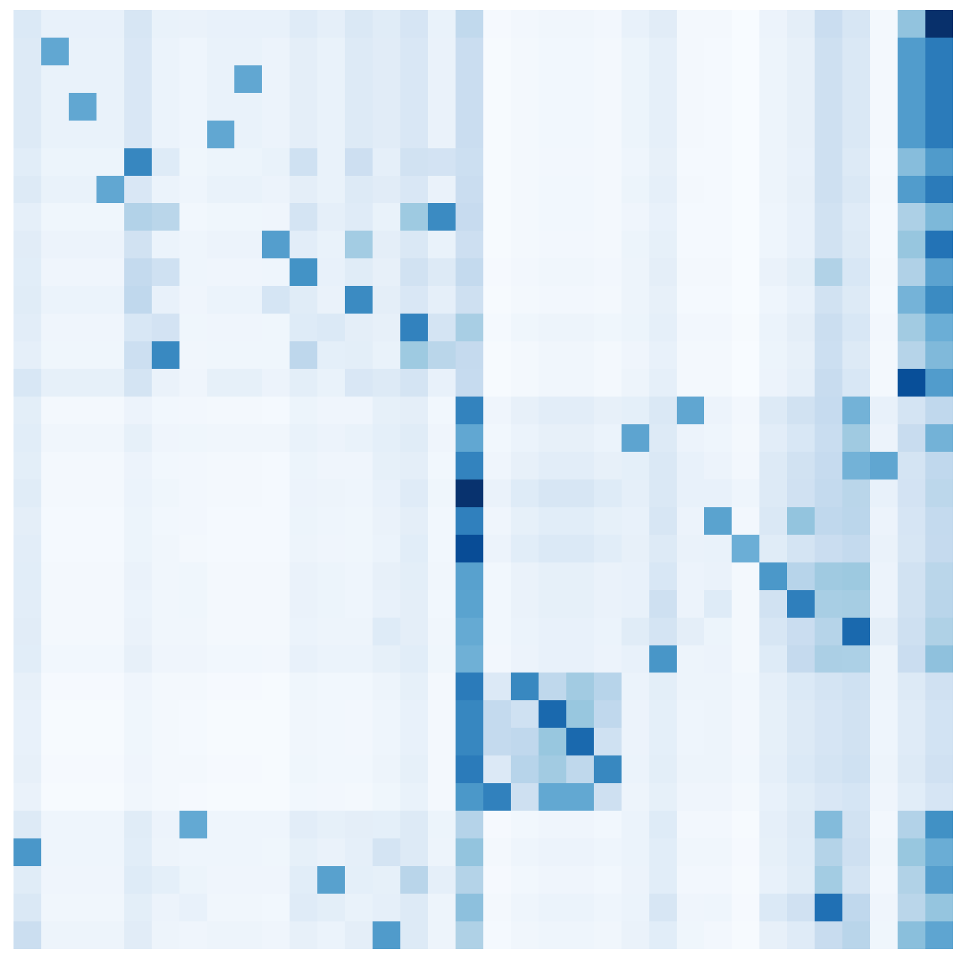
We use node embeddings to avoid slow, domain-specific graph algorithms for node classification, semi-supervised clustering, and anomaly detection.

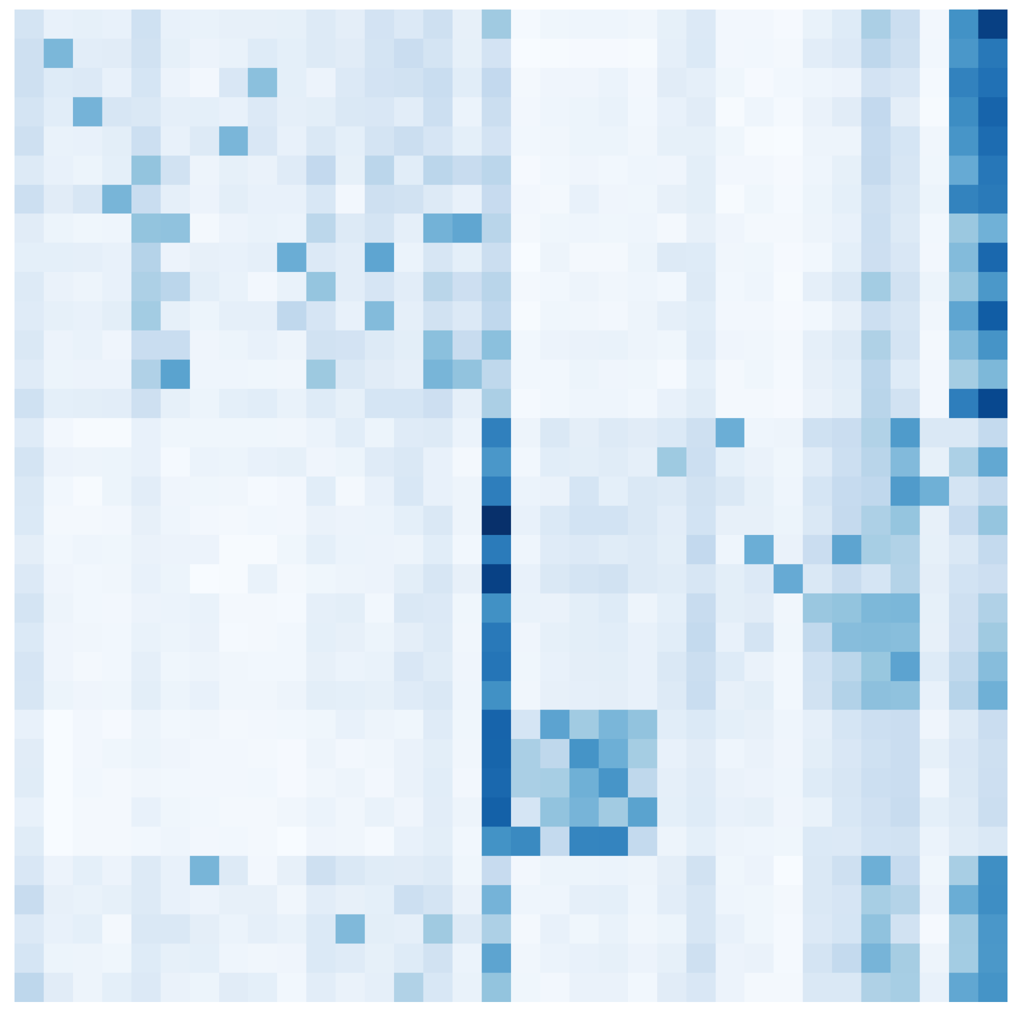


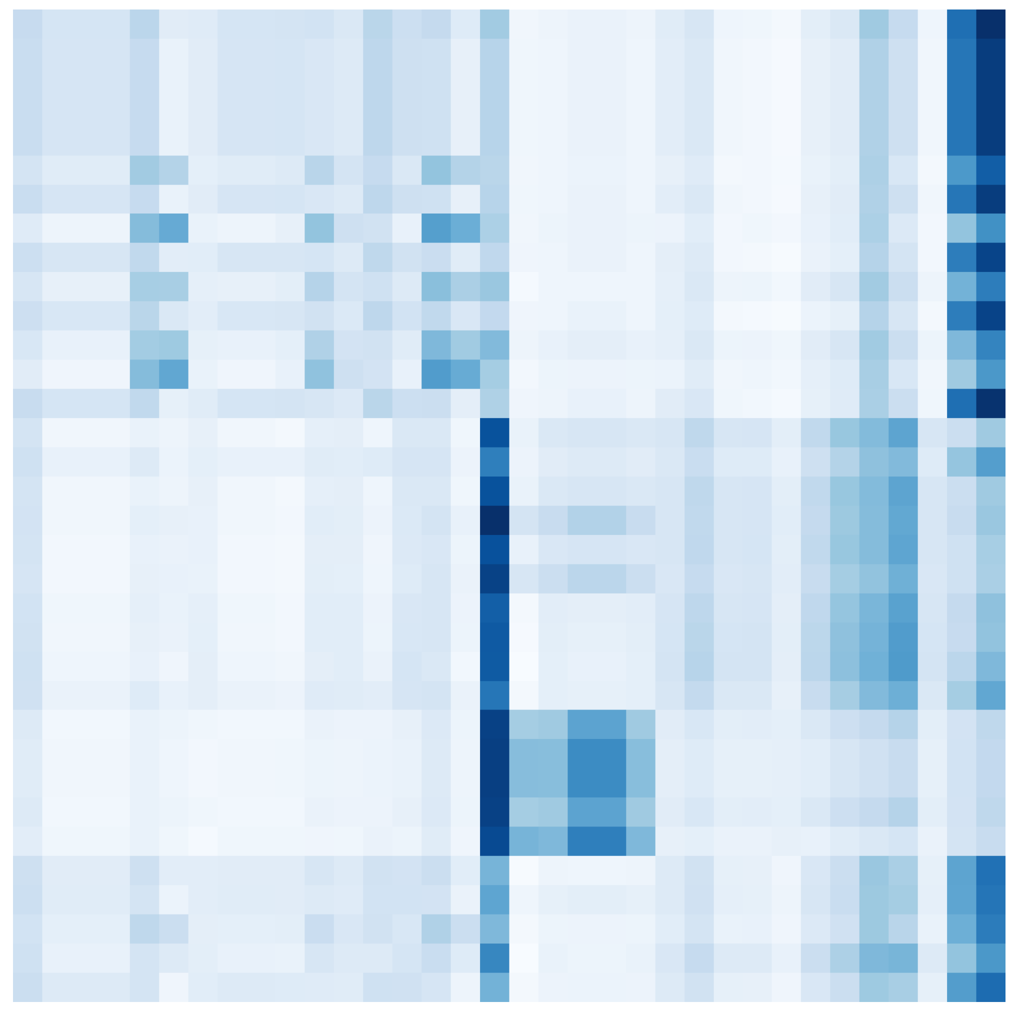
Chapter 4: Versatile node embeddings

Core idea:

We can convert node similarity distributions that allow sampling into representations in the Euclidean space with a single-layer neural network.


Similarity


VERSE


SVD

Chapter 4: Versatile node embeddings

Core idea:

We can convert node similarity distributions that allow sampling into representations in the Euclidean space with a single-layer neural network.

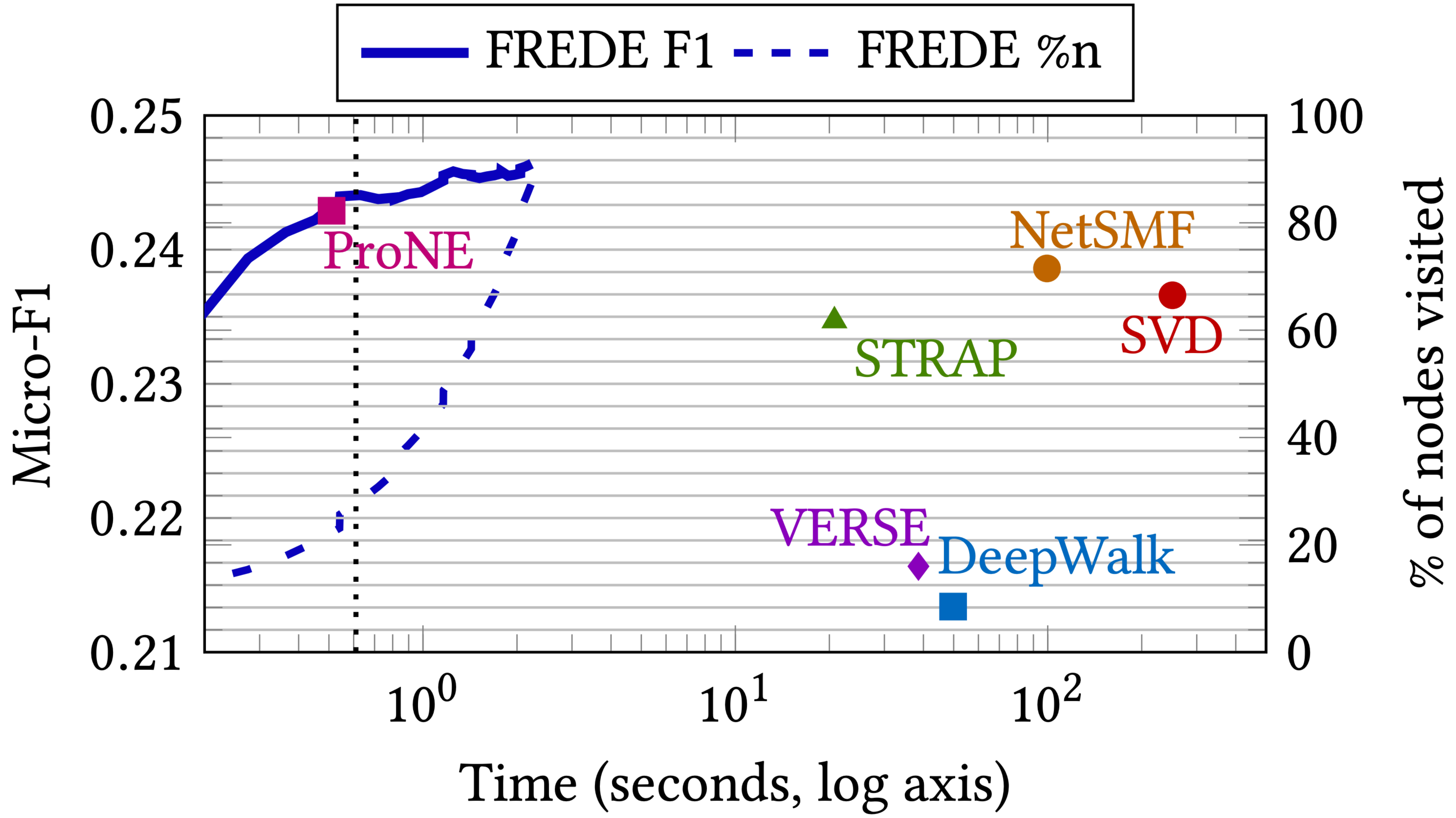
Contributions:

- Time and space complexities of VERSE are $\mathcal{O}(n)$ and $\mathcal{O}(m)$ compared to $\mathcal{O}(n \log n)$ or $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^3)$ space complexity of the previous work.
- Interpretation of previous work in terms of personalised PageRank.
- State-of-the-art results in node classification and link prediction, independently verified in (Mara *et al.*, 2020)

Chapter 5: Anytime node embeddings

Core idea:

Neural embedding models implicitly performs matrix factorisation. Instead of factorising, we sketch a matrix to achieve provable covariance error in linear time.



Chapter 5: Anytime node embeddings

Core idea:

Neural embedding models implicitly performs matrix factorisation. Instead of factorising, we sketch a matrix to achieve provable covariance error in linear time.

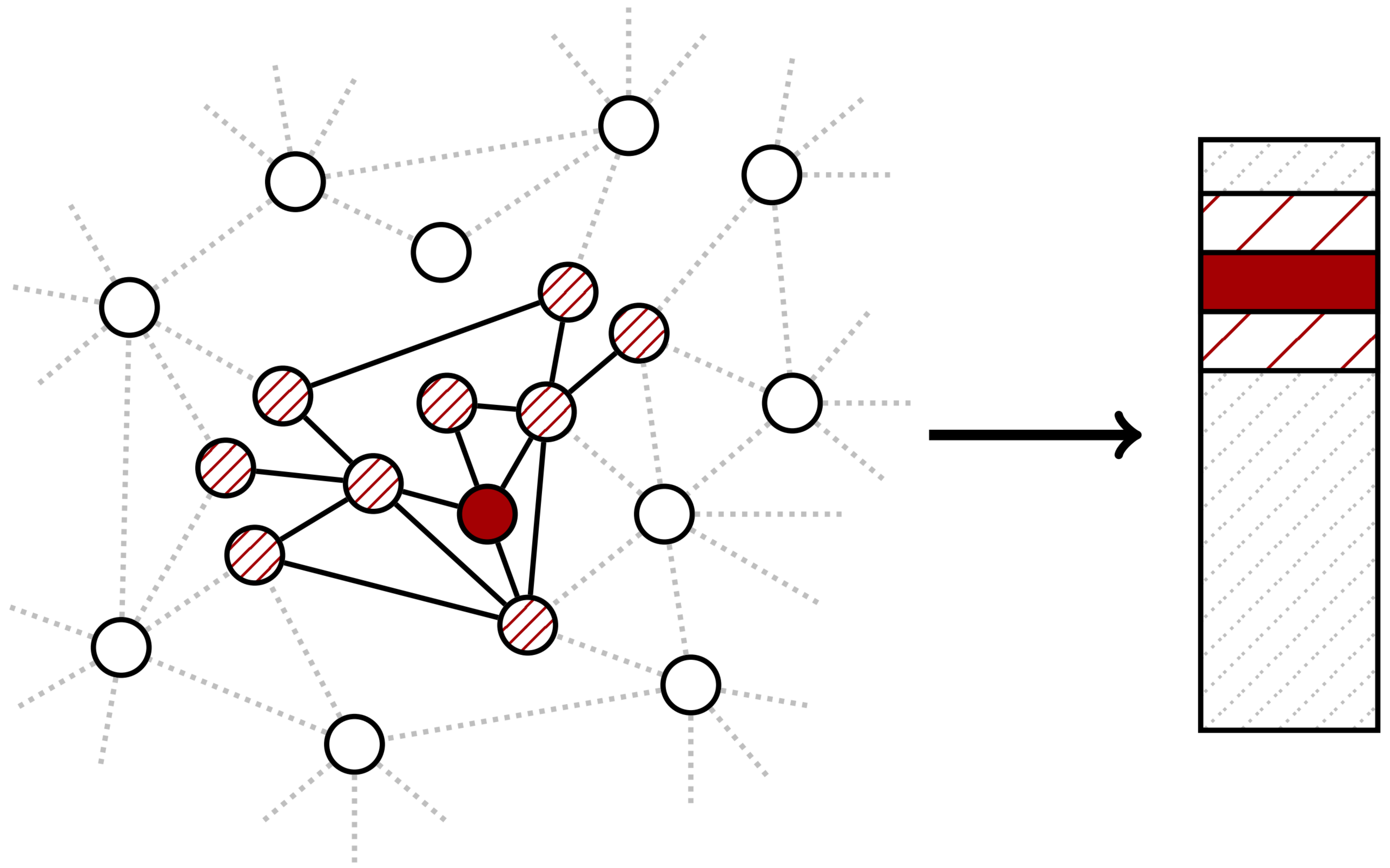
Contributions:

- We analyse the optimal solution of VERSE, and interpret its solution as matrix factorisation.
- Instead of $\mathcal{O}(n^3)$ factorisation, sketch the matrix row-by-row, with guarantees on the solution. Time complexity is $\mathcal{O}(n)$ amortised per row. The solution can be retrieved at **any time**.
- Competitive results in quality on some datasets with processing less that 1% of nodes!

Chapter 6: Local node embeddings

Core idea:

Hashing a similarity row is a local operation; a local similarity, such as approximate personalised PageRank, yields a first local embedding algorithm.



Chapter 6: Local node embeddings

Core idea:

Hashing a similarity row is a local operation; a local similarity, such as approximate personalised PageRank, yields a first local embedding algorithm.

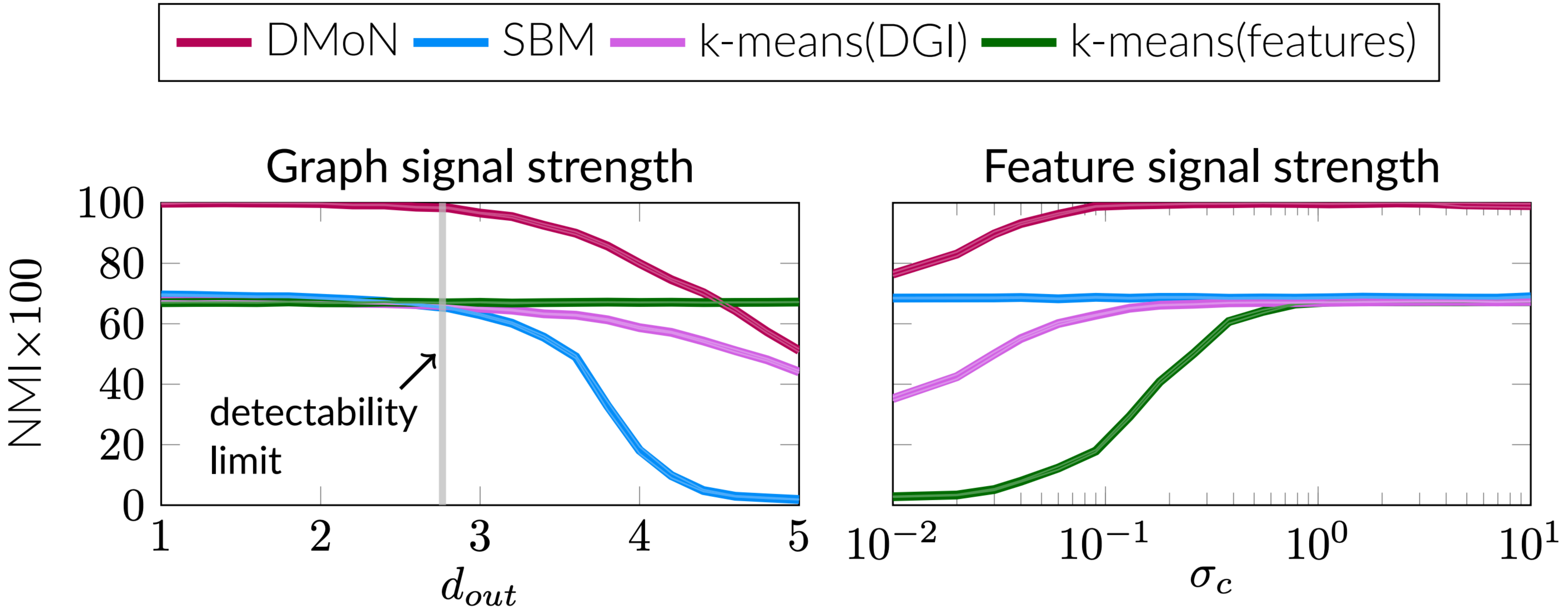
Contributions:

- We formulate the requirements for a local embedding algorithm to be useful in practice.
- We prove that the time complexity of our algorithm is sublinear to the number of graph nodes.
- Experimentally, our algorithm achieves similar or better performance at the time cost 9,000× less than the fastest baseline. Our algorithm is also trivially distributable and parallelizable.

Chapter 7: Deep modularity networks

Core idea:

In attributed graphs, no explicit similarity can capture different signal strength from graph structure and node features. We use a spectral form of the modularity clustering as a differentiable objective function for graph neural network.



Chapter 7: Deep modularity networks

Core idea:

In attributed graphs, no explicit similarity can capture different signal strength from graph structure and node features. We use a spectral form of the modularity clustering as a differentiable objective function for graph neural network.

Contributions:

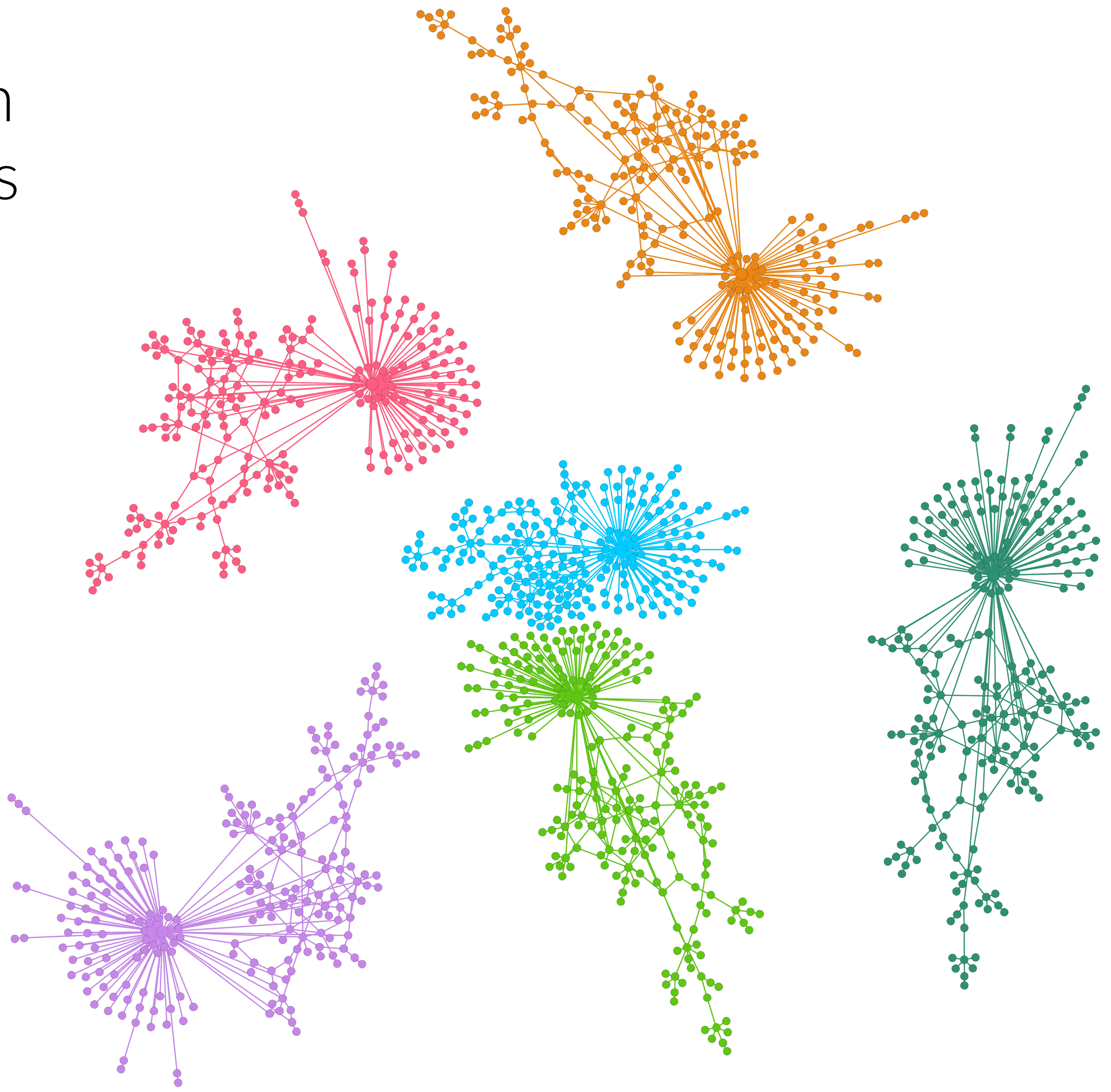
- We formulate a first end-to-end graph clustering method utilising graph neural networks and develop a novel regularizer that empirically allows for better optimization.
- We develop a novel way of benchmarking graph neural networks (published at GLB workshop).
- Empirically, we outperform competing neural network methods by up to 50% in terms of graph metrics and ground truth label correlation.

Introduction to graph representations

In real world, we have graphs with >1 billion edges. We also have millions of small graphs that need to be represented.

Isomorphism or edit distance computations are impractical on graphs with >100 nodes. Kernels are not suited for large collections.

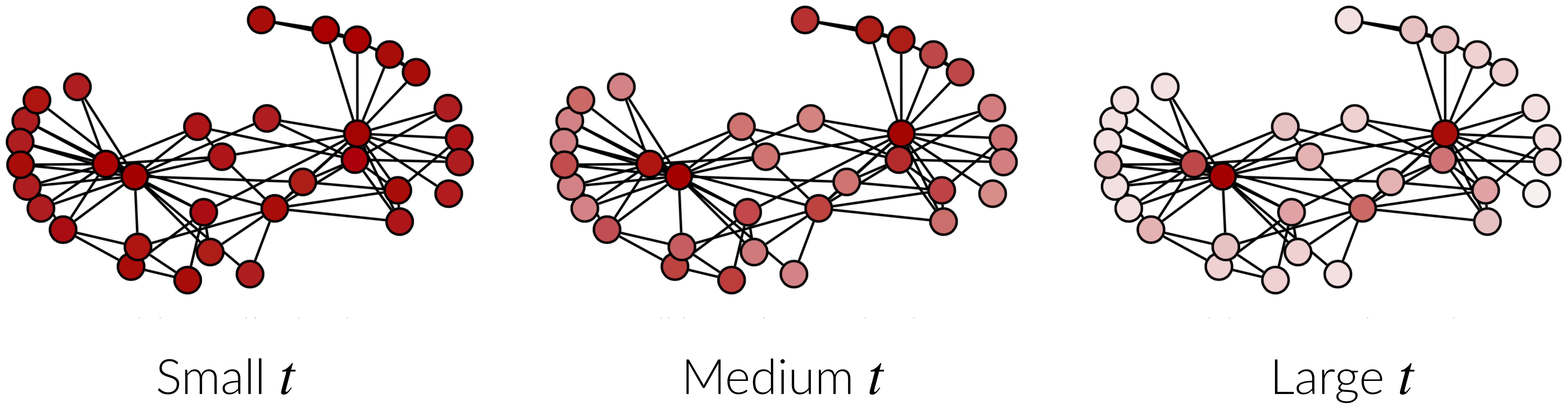
Thus, we need scalable algorithms for embedding graphs in the Euclidean space.



Chapter 9: Spectral graph similarity

Core idea:

Heat kernel trace is an easy-to-compute lower bound to Gromov–Wasserstein distance. We can use the trace of a heat kernel as an embedding of a graph.



Chapter 9: Spectral graph similarity

Core idea:

Heat kernel trace is an easy-to-compute lower bound to Gromov–Wasserstein distance. We can use the trace of a heat kernel as an embedding of a graph.

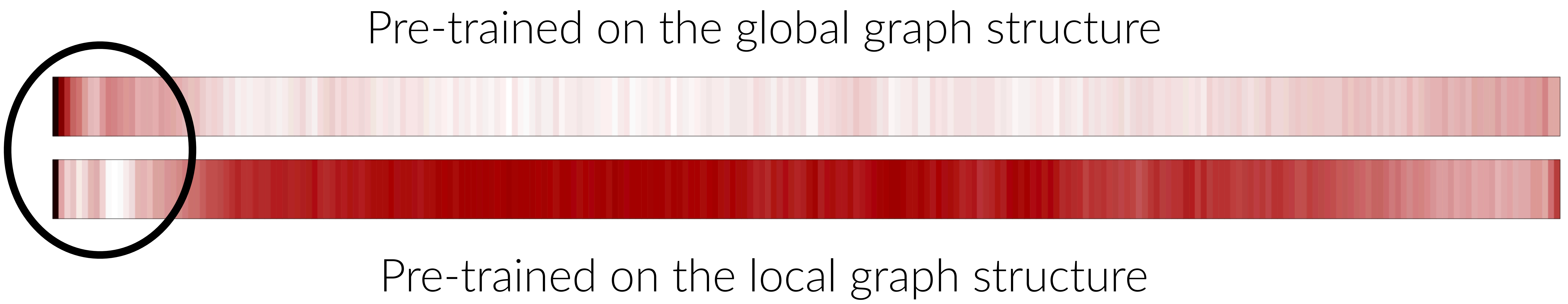
Contributions:

- NetLSD is the first multi–scale graph comparison method that works on unaligned graphs. The method is also optionally size–invariant.
- We propose two approximation schemes that bring down the time complexity of the computation from $\mathcal{O}(n^3)$ to $\mathcal{O}(m)$ without impacting the downstream task quality.
- We show that NetLSD significantly outperforms its less scalable counterparts on real and synthetic graphs in a variety of tasks.

Chapter 10: Learned graph similarity

Core idea:

Heat kernel trace is a set of functions applied to the spectrum of the graph.
Can we train a self-supervised neural network to learn these functions?



Chapter 10: Learned graph similarity

Core idea:

Heat kernel trace is a set of functions applied to the spectrum of the graph.
Can we train a self-supervised neural network to learn these functions?

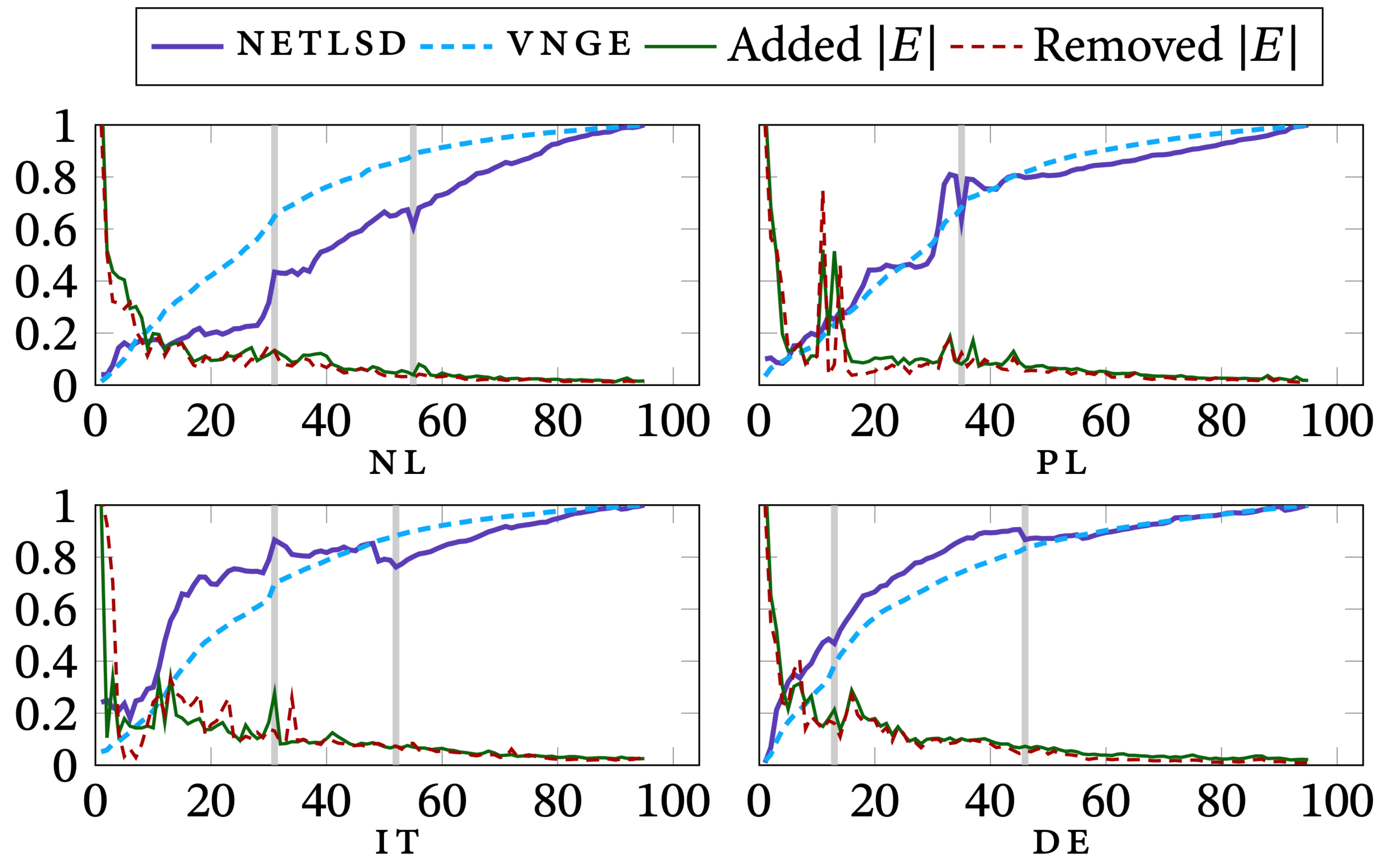
Contributions:

- The first self-supervised method for learning graph representations. Next one is 2020!
- We proposed to use two tasks for pre-training: one local and one global.
- In an extensive set of experiments we demonstrate that no one-size-fits-all solution is possible.

Chapter 11: Approximating spectral similarities

Core idea:

Spectral functions in NetLSD can be approximated with the state-of-the-art numerical linear algebra techniques.



Chapter 11: Approximating spectral similarities

Core idea:

Spectral functions in NetLSD can be approximated with the state-of-the-art numerical linear algebra techniques.

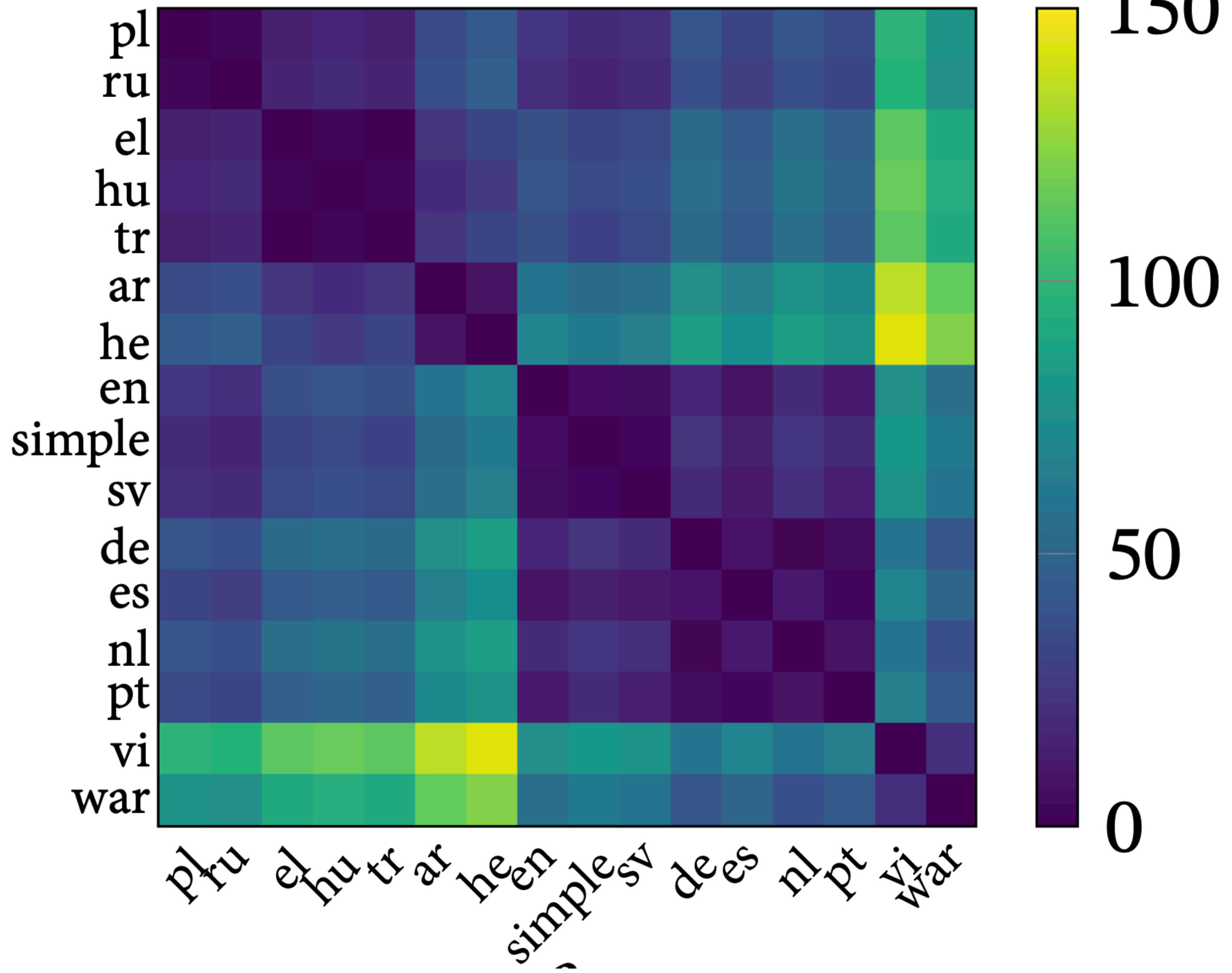
Contributions:

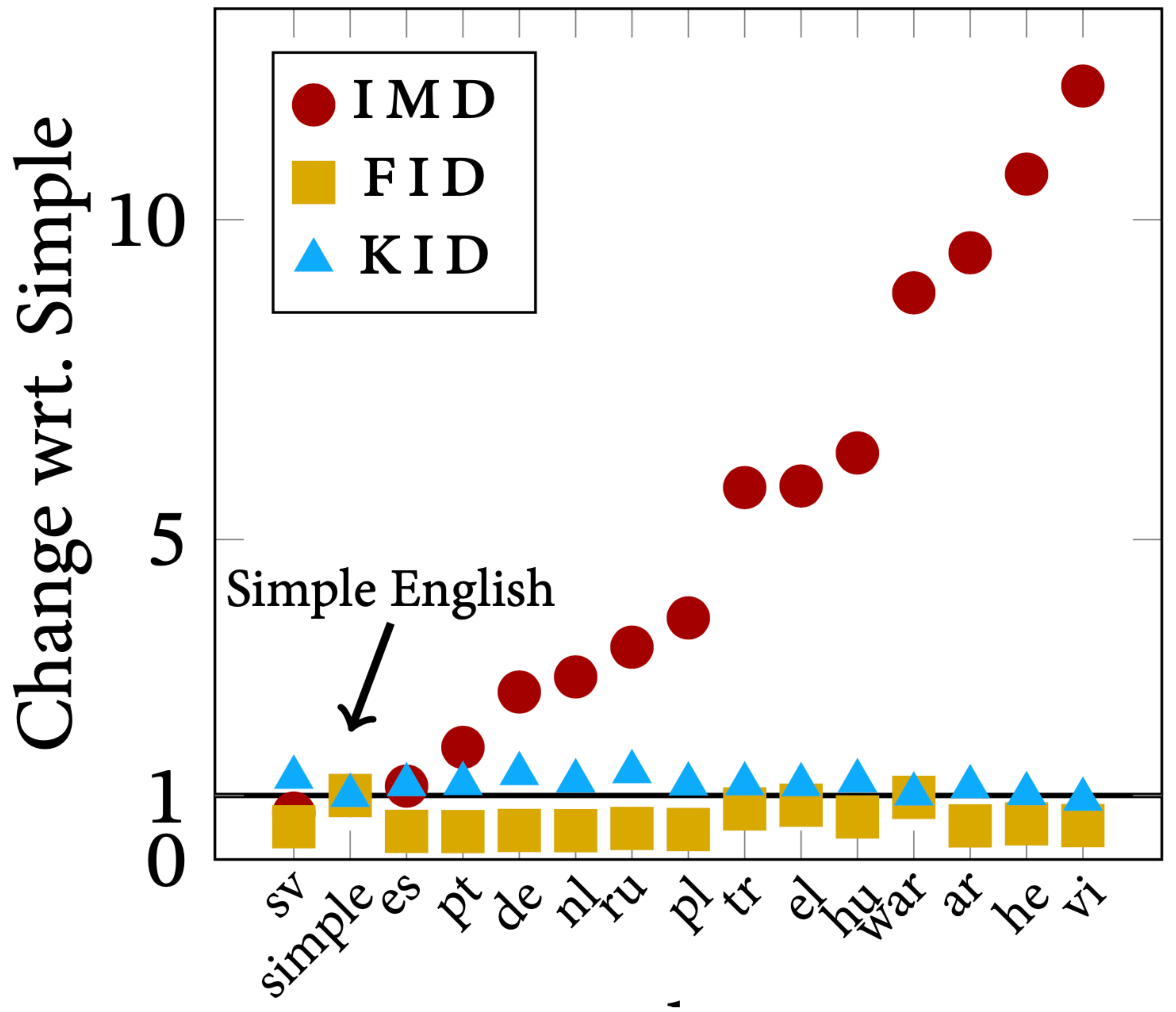
- SLaQ is the first distance function to scale to billion-node graphs.
- We derive the error bounds for NetLSD and show how to support other spectral functions.
- Experimentally, approximation error is 20–200× less than other methods, including ones proposed in Chapter 9. The computations are faster than some naïve baselines.

Chapter 12: Comparing data distributions with graph-based similarities

Core idea:

Laplacian of the k-NN graph approximates the Laplacian of the data manifold. We can then leverage NetLSD and SLaQ to provide a scalable distance measure for data distributions that can even lie in spaces of different dimensionality.





Chapter 12: Comparing data distributions with graph-based similarities

Core idea:

Laplacian of the k-NN graph approximates the Laplacian of the data manifold. We can then leverage NetLSD and SLaQ to provide a scalable distance measure for data distributions that can even lie in spaces of different dimensionality.

Contributions:

- IMD is the first method to compare unaligned data manifolds.
- It is also the first multi-scale metric for generative adversarial networks (GANs).
- We demonstrate IMD's applicability on a broad set of tasks, including introspection into transformations induced by neural networks, distances between languages through the lens of their words, and generative adversarial network model evaluation.

Contribution summary

Representations of nodes

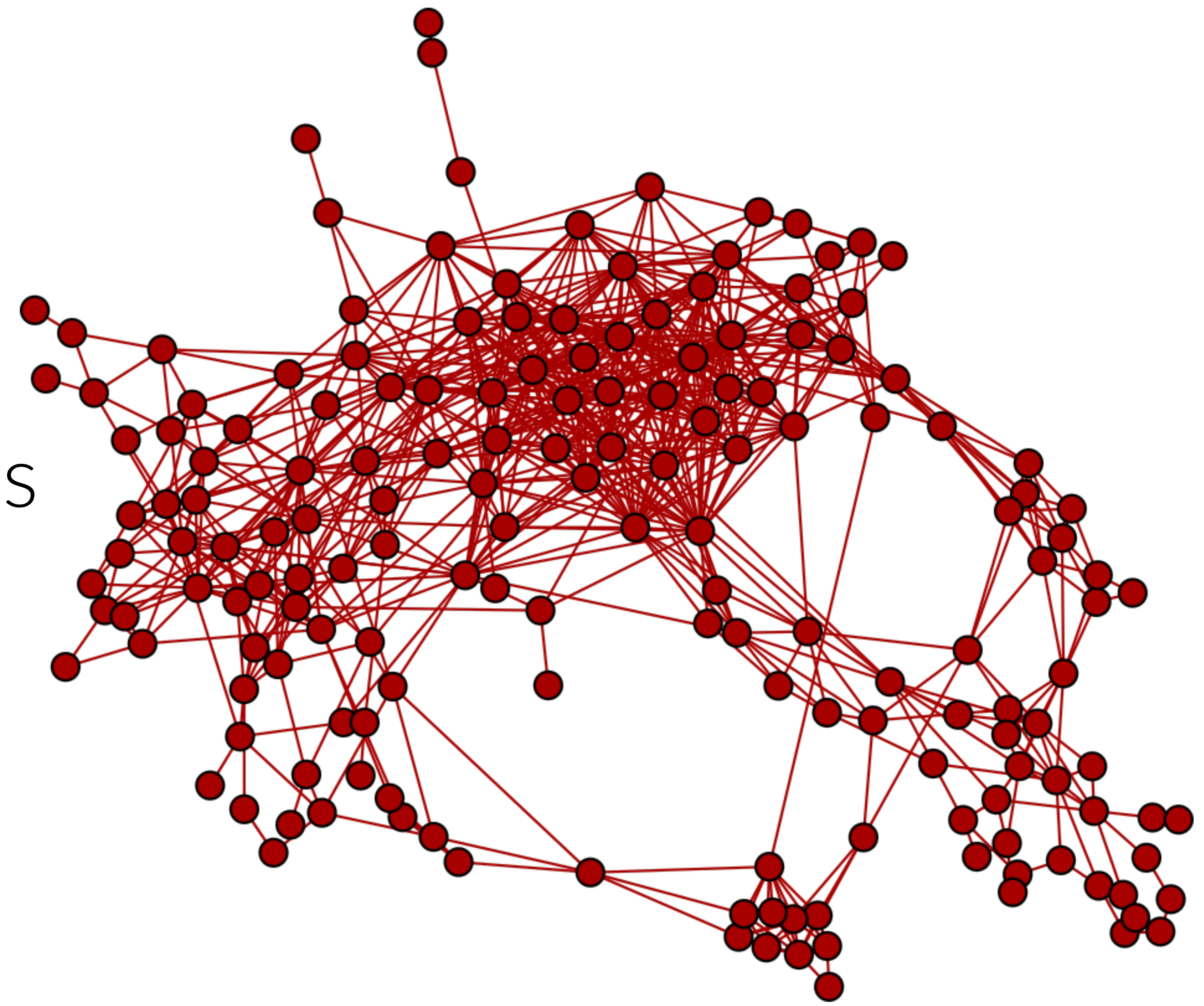
- First local, first anytime algorithms
- Scalability to orders of magnitude larger graphs than reported before, even by commercial systems
- A new paradigm for learning embeddings through graph clustering

Representations of graphs

- First multi-scale, first self-supervised representation-based distances
- Scalable to unaligned graphs with billion+ nodes on a single server in 15m
- Broad applications in machine learning and graph mining

Deeper dive: node embeddings

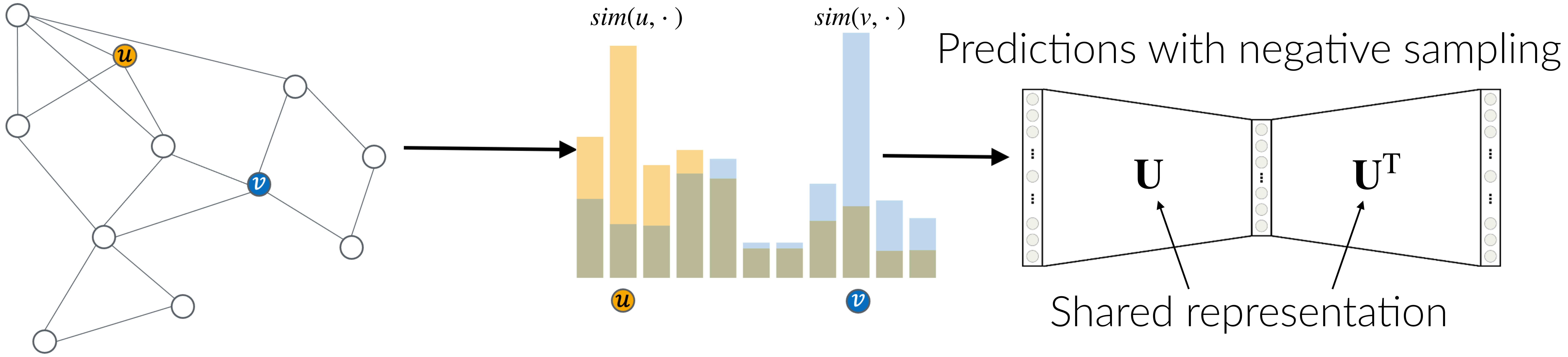
An in-depth review of proposed techniques



Versatile node embeddings

Core idea: abstract node similarities away from the representation framework.

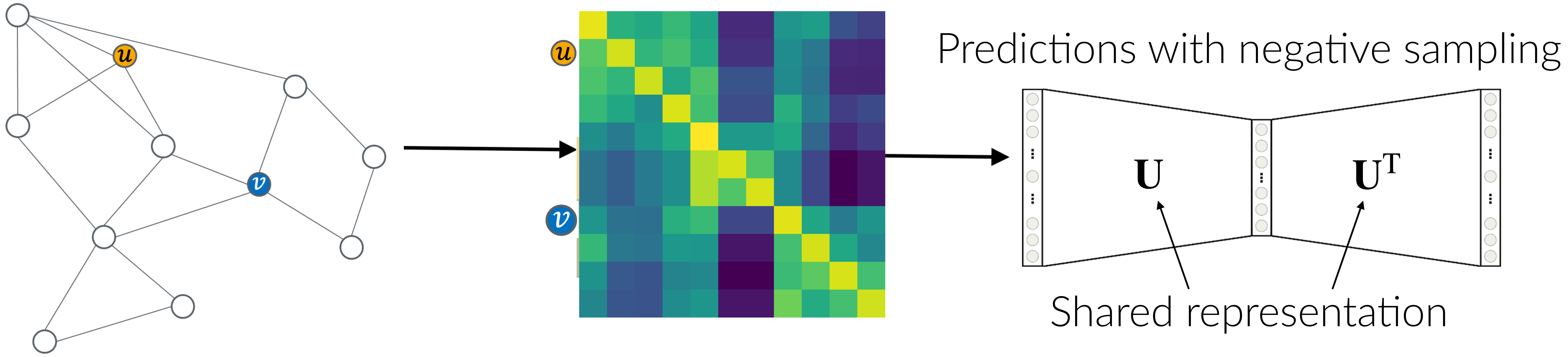
Sample nodes from similarity **distributions**



Versatile node embeddings

Core idea: abstract node similarities away from the representation framework.

Sample nodes from similarity **distributions**

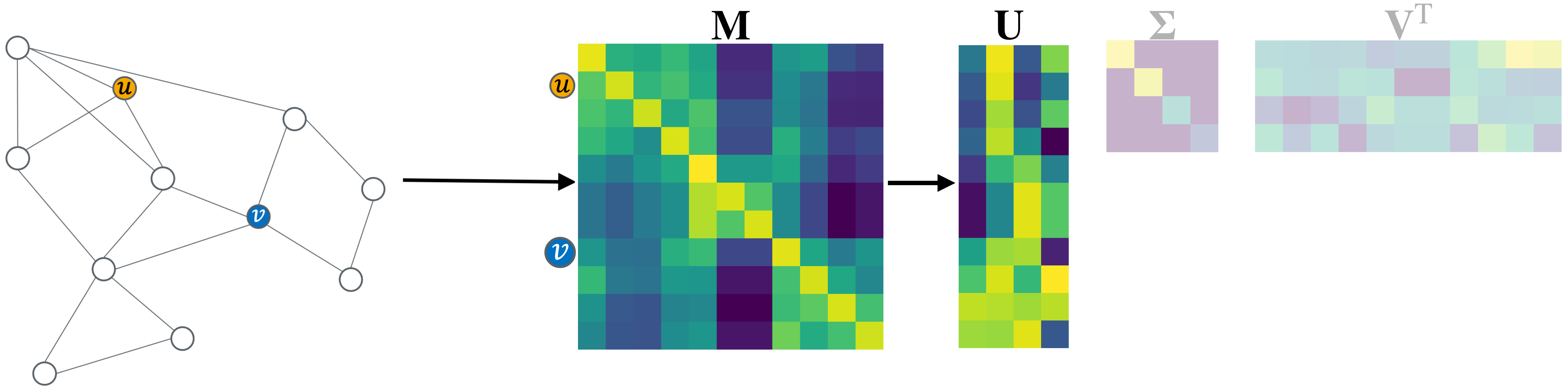


Full similarity matrix $\mathbf{M}$ is never materialised

Update $\mathbf{U}$ with gradient descent

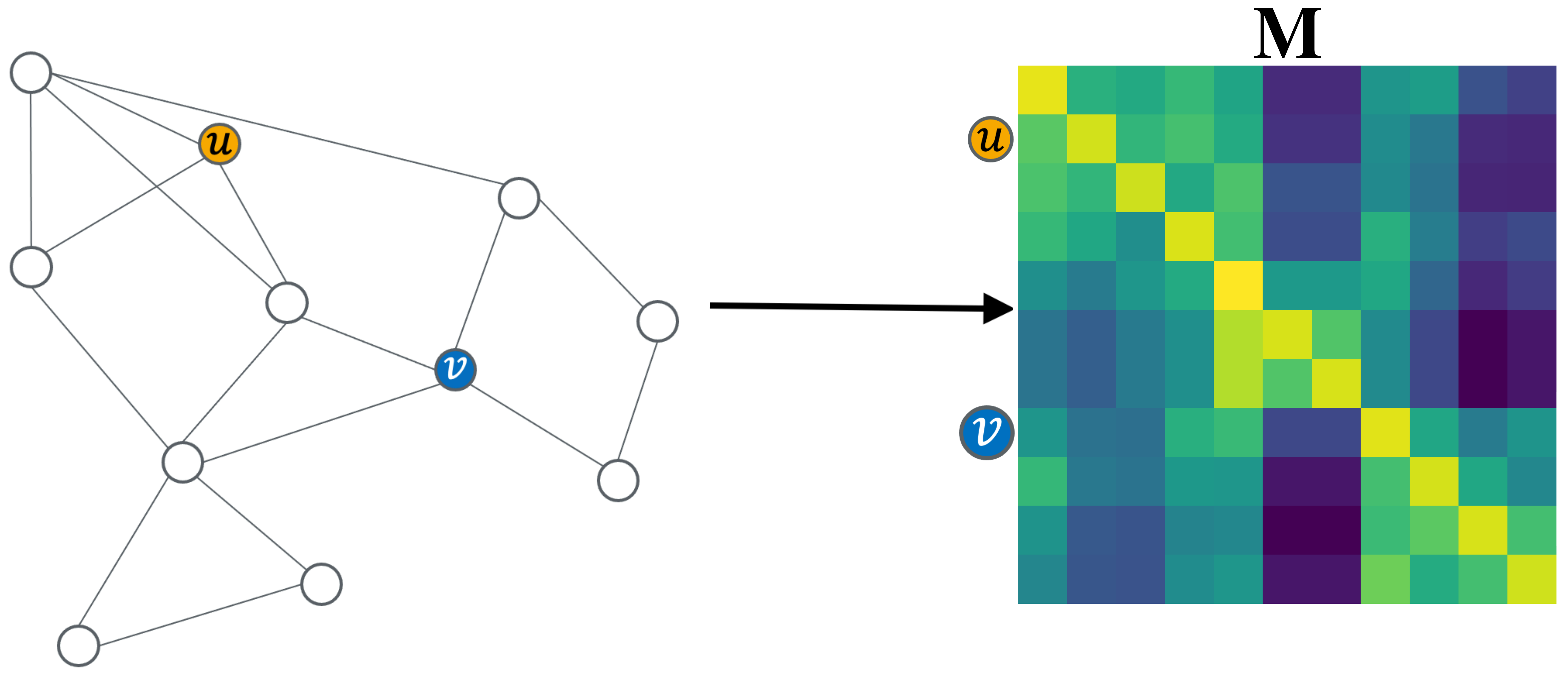
Anytime node embeddings

Core idea: we do not need to waste resources to compute $\mathbf{V}^\mathrm{T}$, and $\mathbf{U}$ can be incrementally **sketched** with embeddings available at any time!



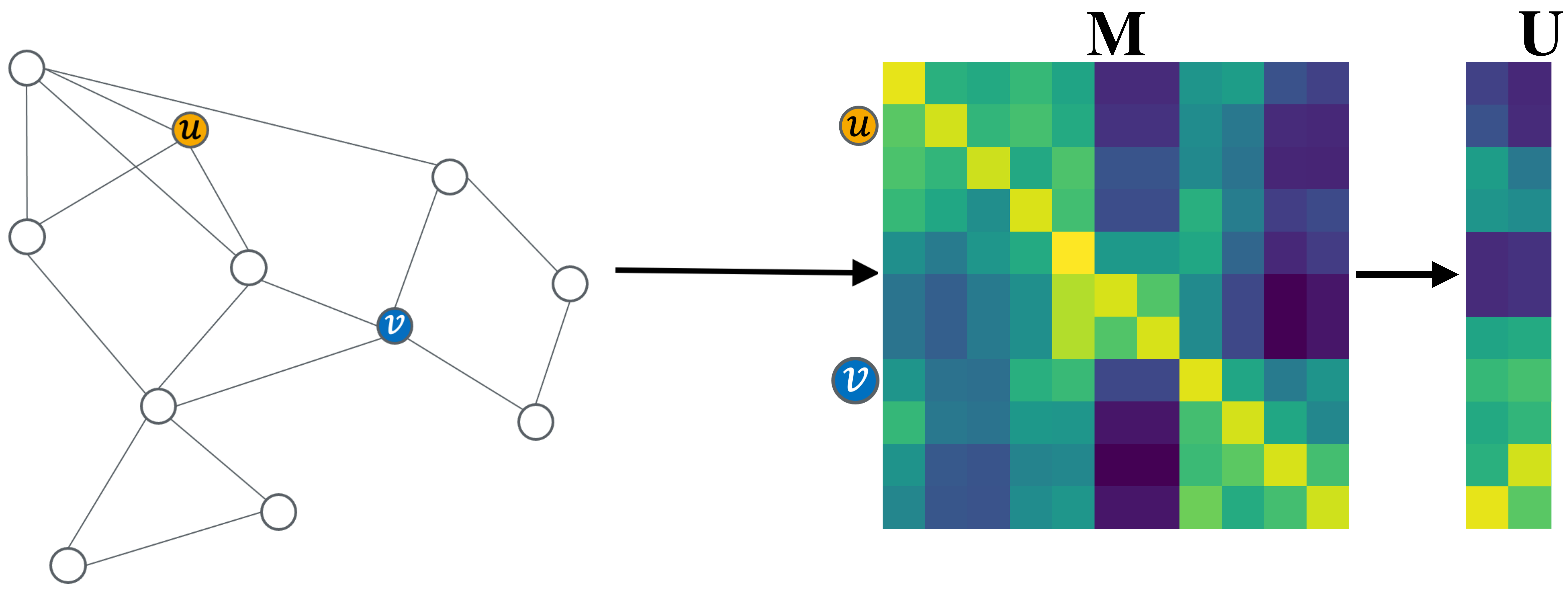
Anytime node embeddings

Core idea: we do not need to waste resources to compute $\mathbf{V}^{\mathrm{T}}$, and $\mathbf{U}$ can be incrementally **sketched** with embeddings available at any time!



Anytime node embeddings

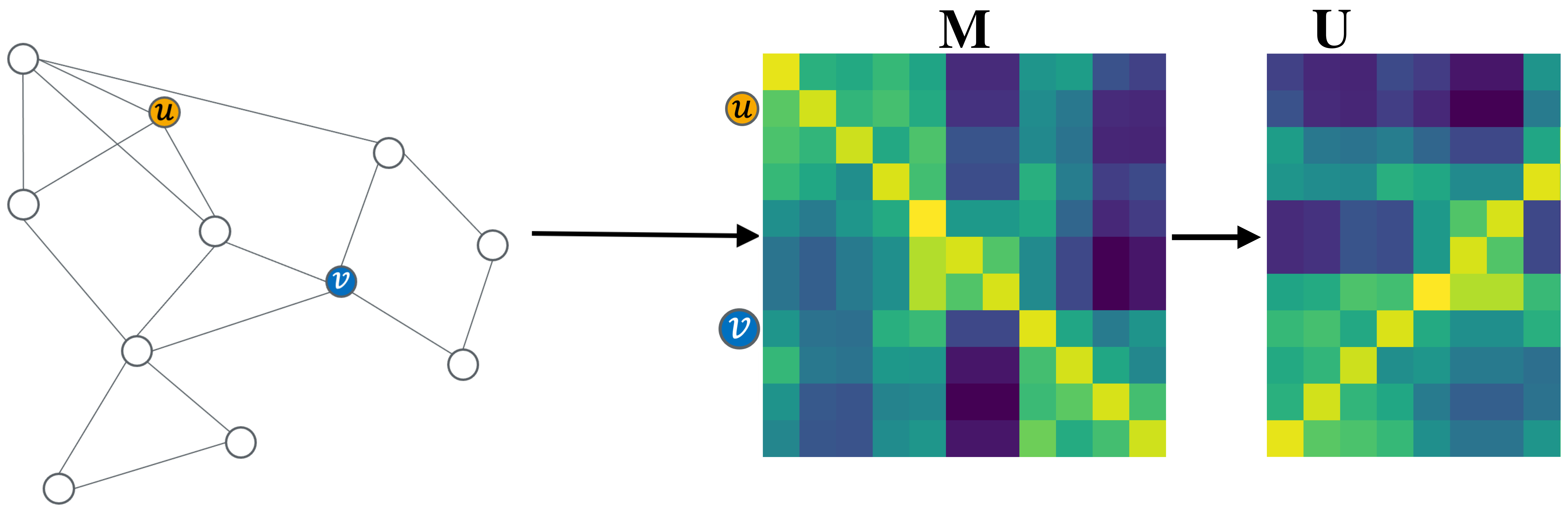
Core idea: we do not need to waste resources to compute $\mathbf{V}^{\mathrm{T}}$, and $\mathbf{U}$ can be incrementally **sketched** with embeddings available at any time!



Insert rows one by one into $\mathbf{U}$

Anytime node embeddings

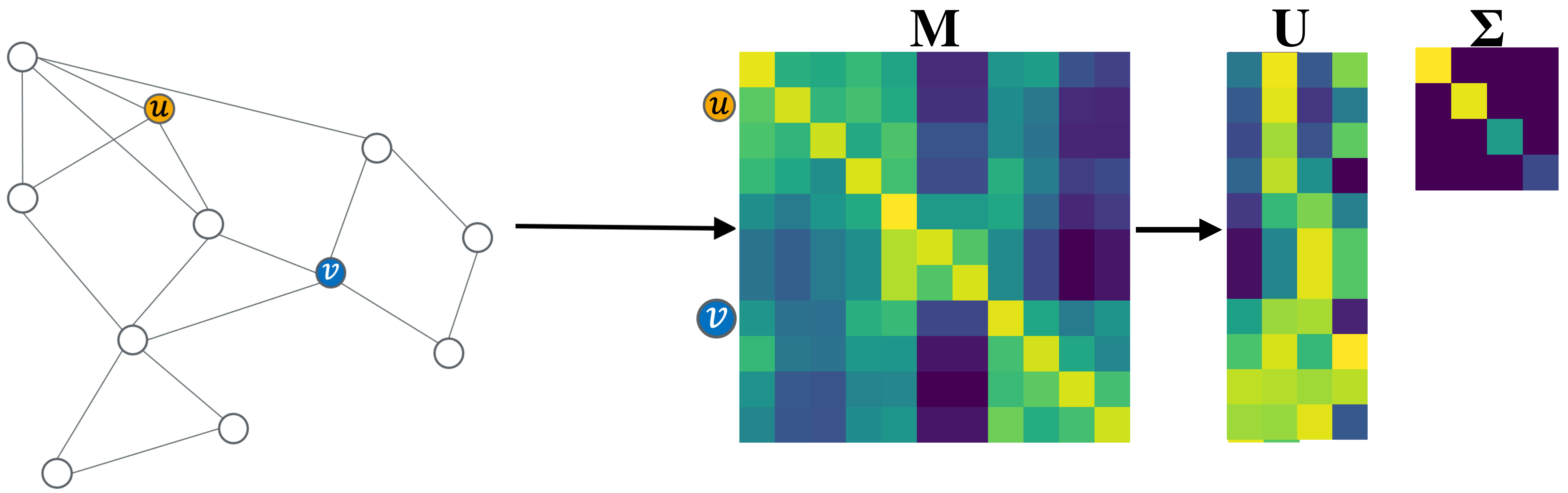
Core idea: we do not need to waste resources to compute $\mathbf{V}^\mathrm{T}$, and $\mathbf{U}$ can be incrementally **sketched** with embeddings available at any time!



Insert rows one by one into $\mathbf{U}$

Anytime node embeddings

Core idea: we do not need to waste resources to compute $\mathbf{V}^{\mathrm{T}}$, and $\mathbf{U}$ can be incrementally **sketched** with embeddings available at any time!

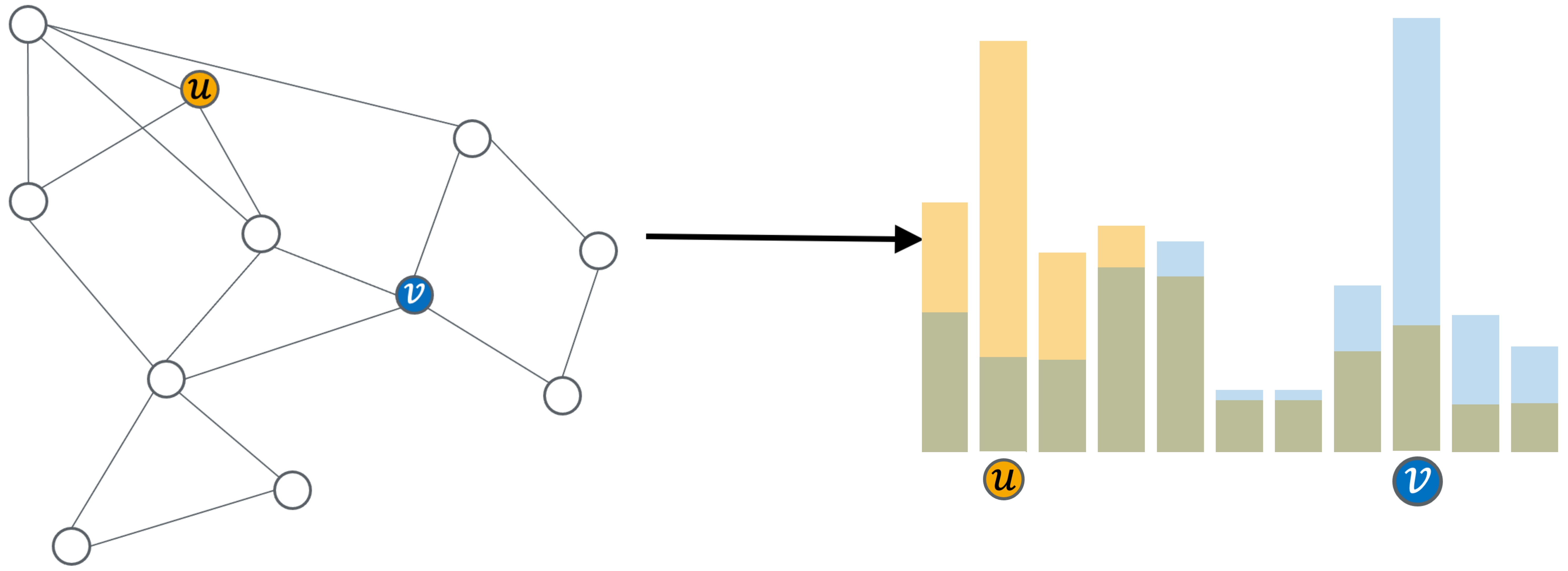


When $\mathbf{U}$ is full, do SVD on it

Retrieve $\mathbf{U}$ at any time!

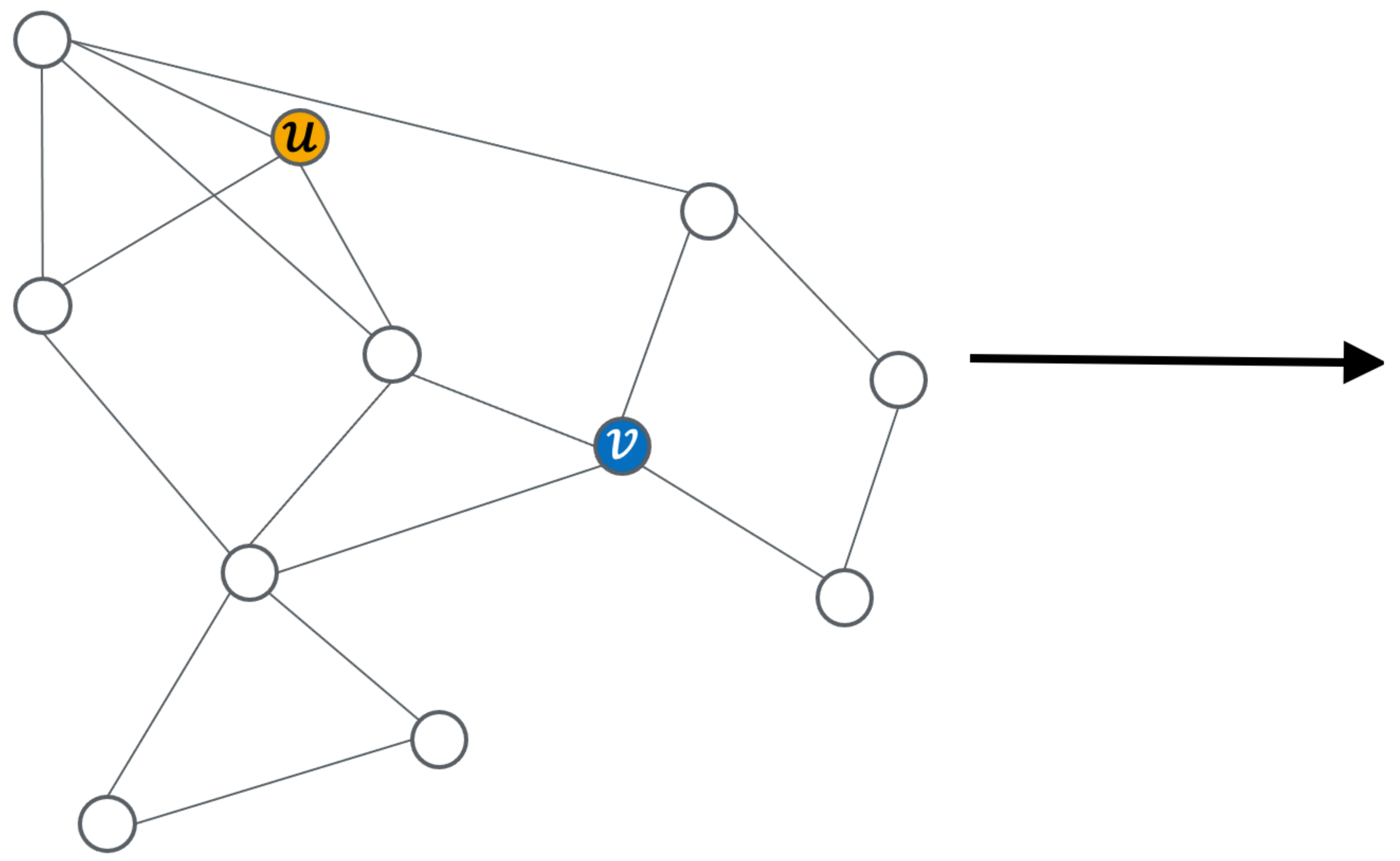
Local node embeddings

Core idea: we do not need to waste **any** resources, let's not compute rows of $\mathbf{U}$ that we will not need! Hash the sparse similarity values into $\mathbf{U}$ directly.



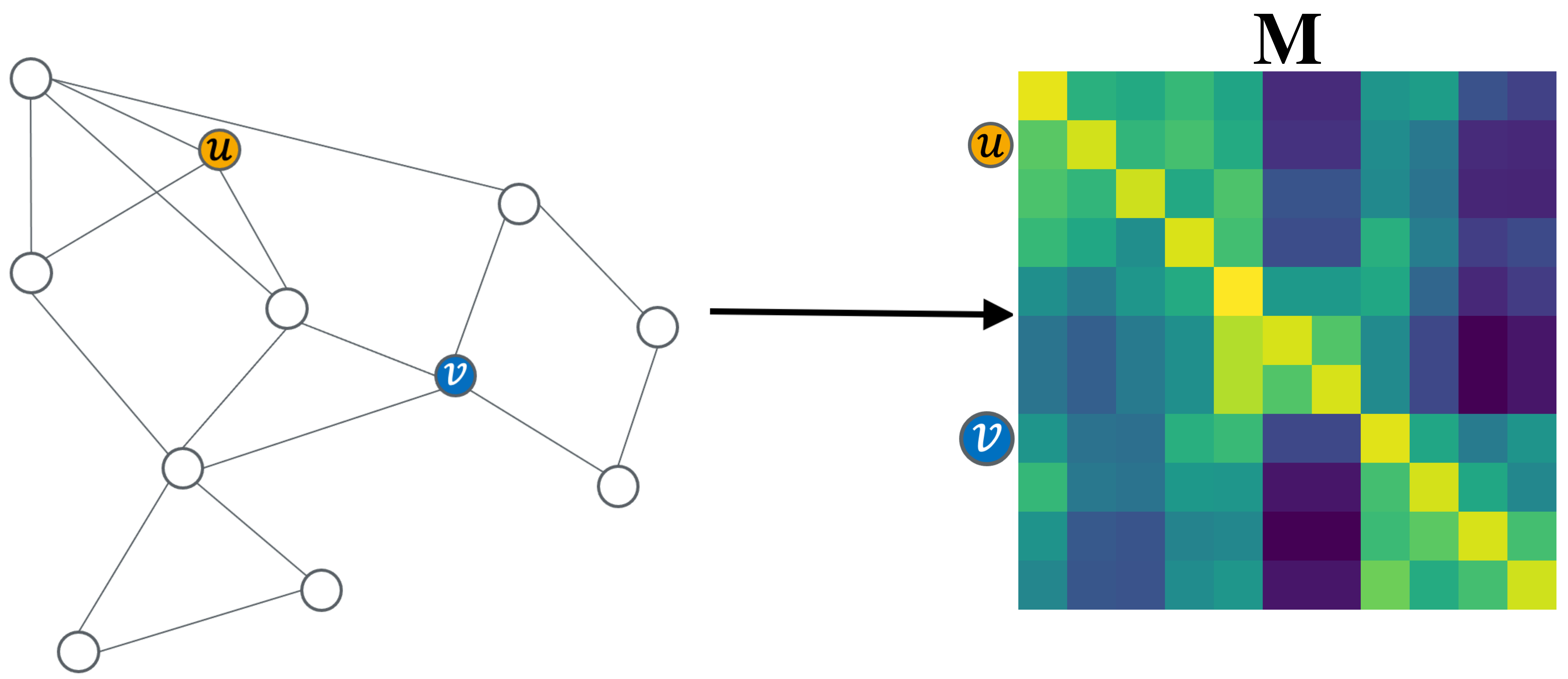
Local node embeddings

Core idea: we do not need to waste **any** resources, let's not compute rows of $\mathbf{U}$ that we will not need! Hash the sparse similarity values into $\mathbf{U}$ directly.



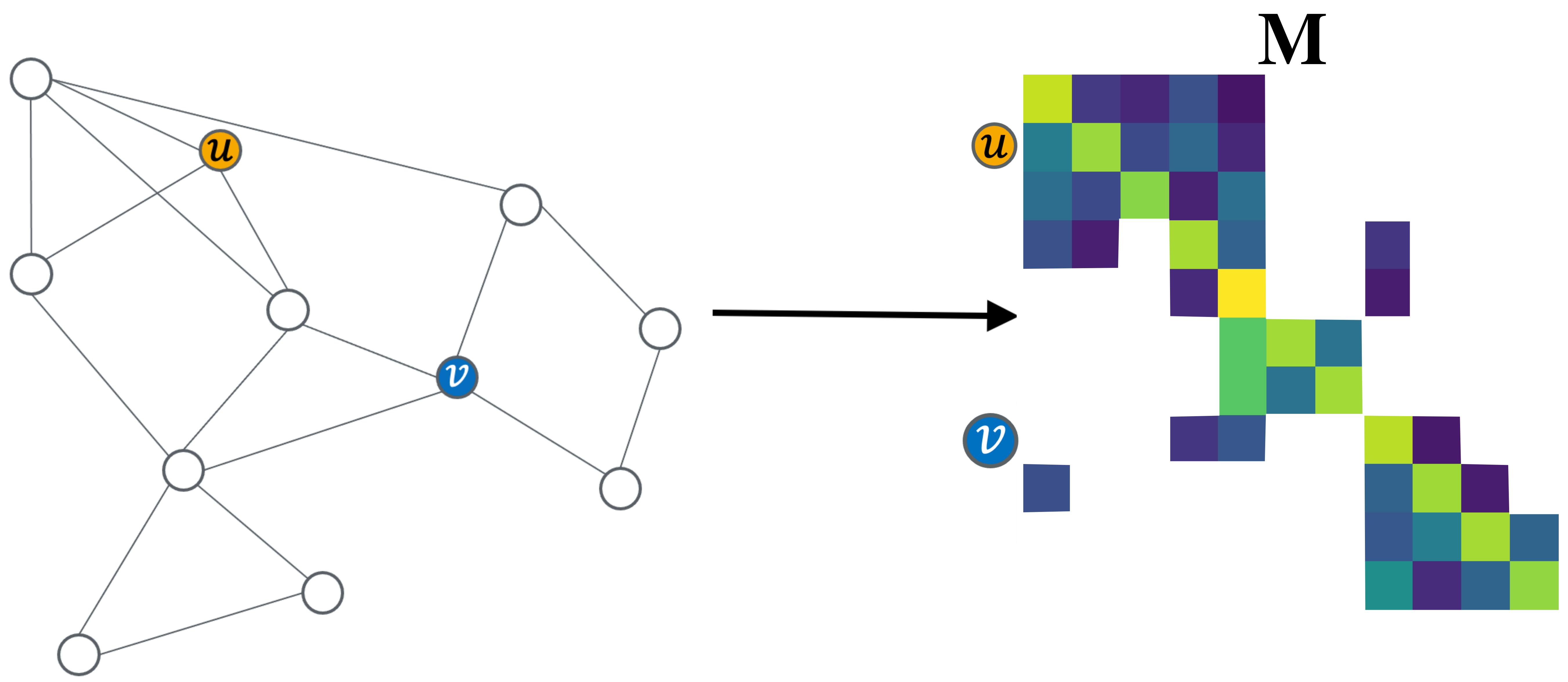
Local node embeddings

Core idea: we do not need to waste **any** resources, let's not compute rows of $\mathbf{U}$ that we will not need! Hash the sparse similarity values into $\mathbf{U}$ directly.



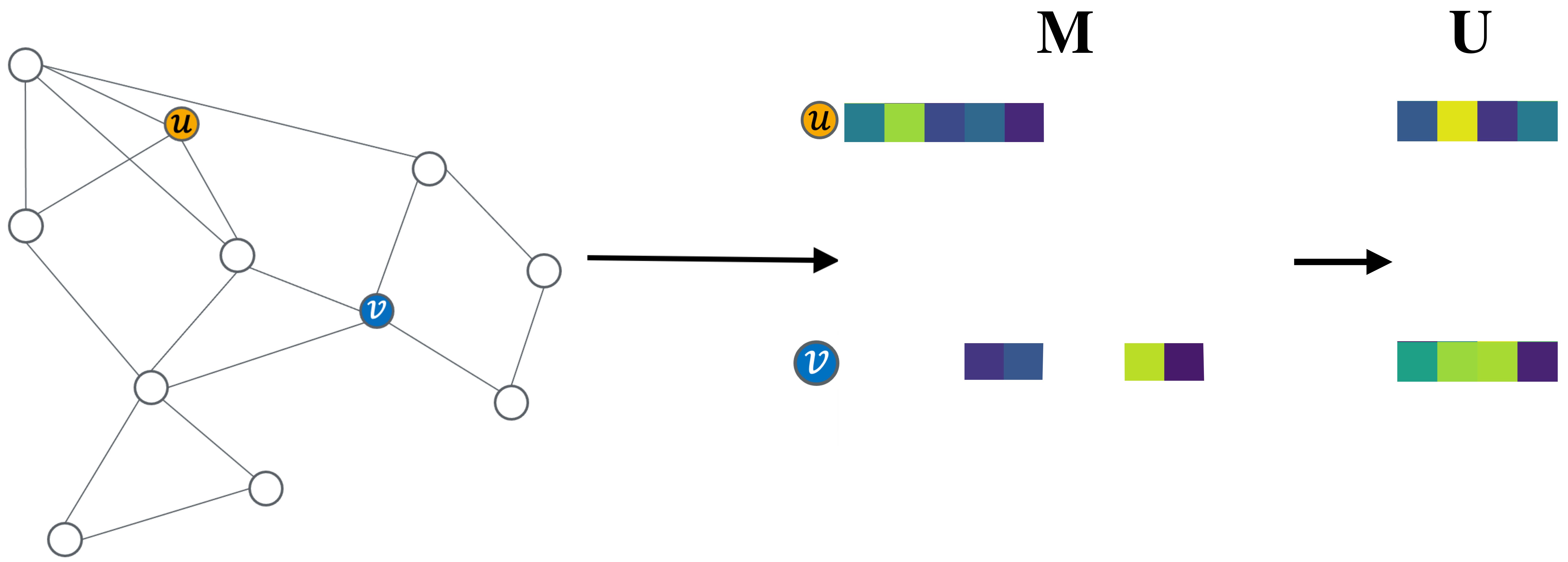
Local node embeddings

Core idea: we do not need to waste **any** resources, let's not compute rows of $\mathbf{U}$ that we will not need! Hash the sparse similarity values into $\mathbf{U}$ directly.



Local node embeddings

Core idea: we do not need to waste **any** resources, let's not compute rows of $\mathbf{U}$ that we will not need! Hash the sparse similarity values into $\mathbf{U}$ directly.



Hash $\mathbf{M}$ as follows: $\mathbf{U}_{v,h(i)} \mathrel{+}= h_{sign}(i) \times \max\left(0, \log \mathbf{M}_{vi}\right)$

Algorithm analysis

Time complexity:

- Reduction from $\mathcal{O}(n \log n)$ to $\mathcal{O}(n)$ time complexity for VERSE
- Reduction from $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space to $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space for FREDE
- Local algorithm's complexity independent of graph size

Guarantees & analysis:

- VERSE shaves off 3 unnecessary parameters of the previous SotA
- FREDE offers fist-of-their-kind guarantees on the covariance of node embeddings
- Local algorithm provably converges to the global solution; proofs of sublinear time & memory

As good as we can get?

As good as we can get?

No.

As good as we can get?

No.

Our local method does not have quality guarantees, only asymptotic convergence.

Approximate similarity computations will lead to other, faster algorithms.

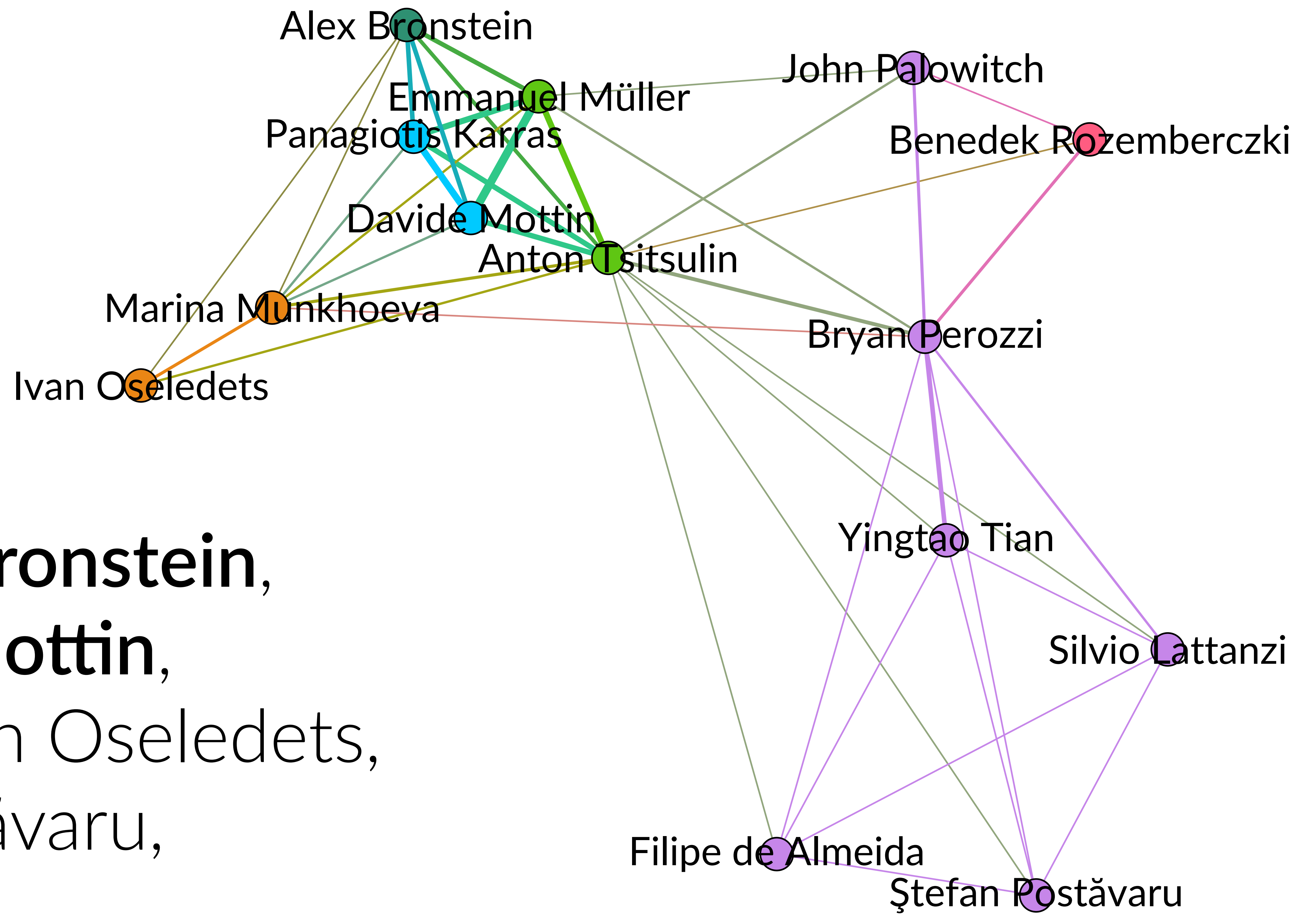
Embeddings of large billion-scale graphs are still unexplored by any other method.

Good luck.

Acknowledgements

13 co-authors, 6 with 3+ papers together

Filipe Miguel Gonçalves de Almeida, **Alex Bronstein**, **Panagiotis Karras**, Silvio Lattanzi, **Davide Mottin**, **Emmanuel Müller**, **Marina Munkhoeva**, Ivan Oseledets, John Palowitch, **Bryan Perozzi**, Ştefan Postăvaru, Benedek Rozemberczki, Yingtao Tian



Summary

Scalable, theoretically grounded algorithms
for building **explicit, expressive** representations
of graph-structured data

Summary

Simplified and explained previous work

Opened new research directions

Scalable, theoretically grounded algorithms
for building **explicit, expressive** representations
of graph-structured data

Open-source

Works in production

Reproduced by others

Spectral graph embeddings

Core idea: heat kernel trace is a very useful representation.

$$\mathbf{H}_t = e^{-t\mathscr{L}} = \mathbf{\Phi} e^{-t\mathbf{\Lambda}} \mathbf{\Phi}^\top = \sum_{j=1}^{n} e^{-t\lambda_j} \phi_j \phi_j^\top$$

$$\mathrm{Tr}(\mathbf{H}_t) = \sum_{j=1}^{n} e^{-t\lambda_j}$$ easy to compute!

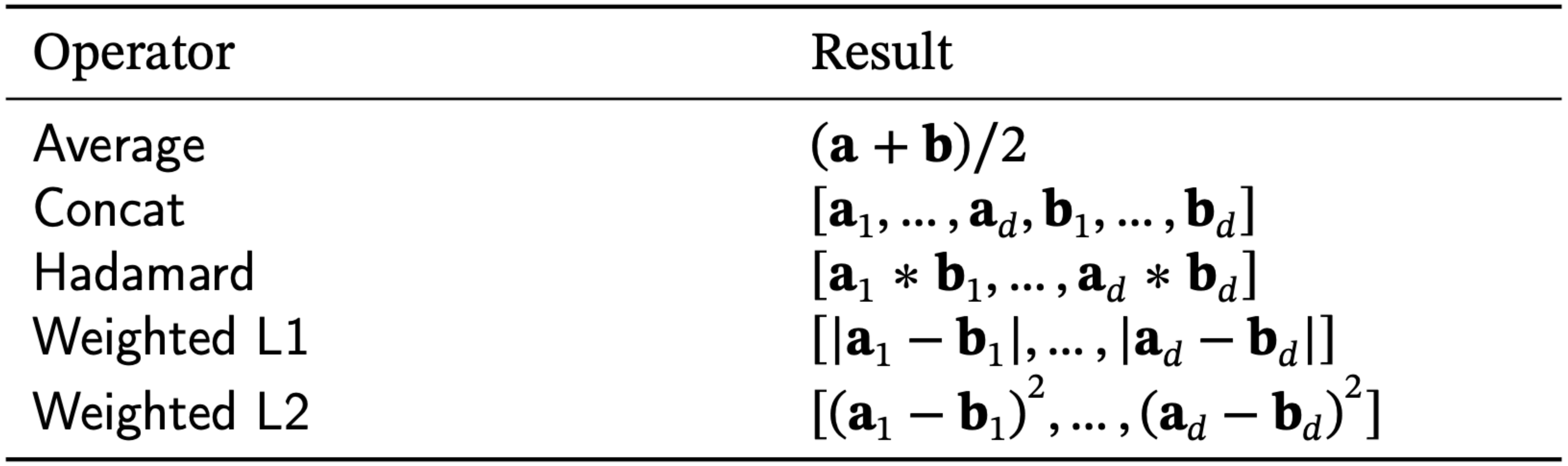
$$d_{GW,p}^{\mathrm{spec}}(X,Y) = \inf_{M} \sup_{t>0} e^{-2\left(t+t^{-1}\right)} \cdot \left(\sum_{i,j} \sum_{i'j'} \left| \mathbf{H}_t^X(x_i, x_{i'}) - \mathbf{H}_t^Y(y_i, y_{i'}) \right|^p m_{ij} m_{i'j'} \right)^{1/p} \geq \sup_{t>0} e^{-2\left(t+t^{-1}\right)} \cdot \left| \mathrm{Tr}\left(\mathbf{H}_t^X\right) - \mathrm{Tr}\left(\mathbf{H}_t^Y\right) \right|$$

Approximate spectral similarities

Core idea: we can approximate trace of matrix functions with stochastic estimation.

Hutchinson trace estimator	$\mathbf{Tr}\left(f(\mathbf{M})\right) = \mathbb{E}\left[z^\top f(\mathbf{M})z\right]$
Gaussian quadrature	$\mathbb{E}\left[z^\top f(\mathbf{M})z\right] \approx \sum_{k=1}^{s} w_k f\left(x_k\right)$
s-step Lanczos algorithm	with weights $\boldsymbol{w_k}$ and points $\boldsymbol{x_k}$

Extra: link prediction from embeddings



Operator	Result
Average	$(\mathbf{a}+\mathbf{b})/2$
Concat	$[\mathbf{a}_1, \dots, \mathbf{a}_d, \mathbf{b}_1, \dots, \mathbf{b}_d]$
Hadamard	$[\mathbf{a}_1 * \mathbf{b}_1, \dots, \mathbf{a}_d * \mathbf{b}_d]$
Weighted L1	$[\lvert\mathbf{a}_1 - \mathbf{b}_1\rvert, \dots, \lvert\mathbf{a}_d - \mathbf{b}_d\rvert]$
Weighted L2	$[(\mathbf{a}_1 - \mathbf{b}_1)^2, \dots, (\mathbf{a}_d - \mathbf{b}_d)^2]$

Table 2.3: Vector operators used for link-prediction task for each $u, v \in V$ and corresponding embeddings $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$.

Extra: link prediction from embeddings

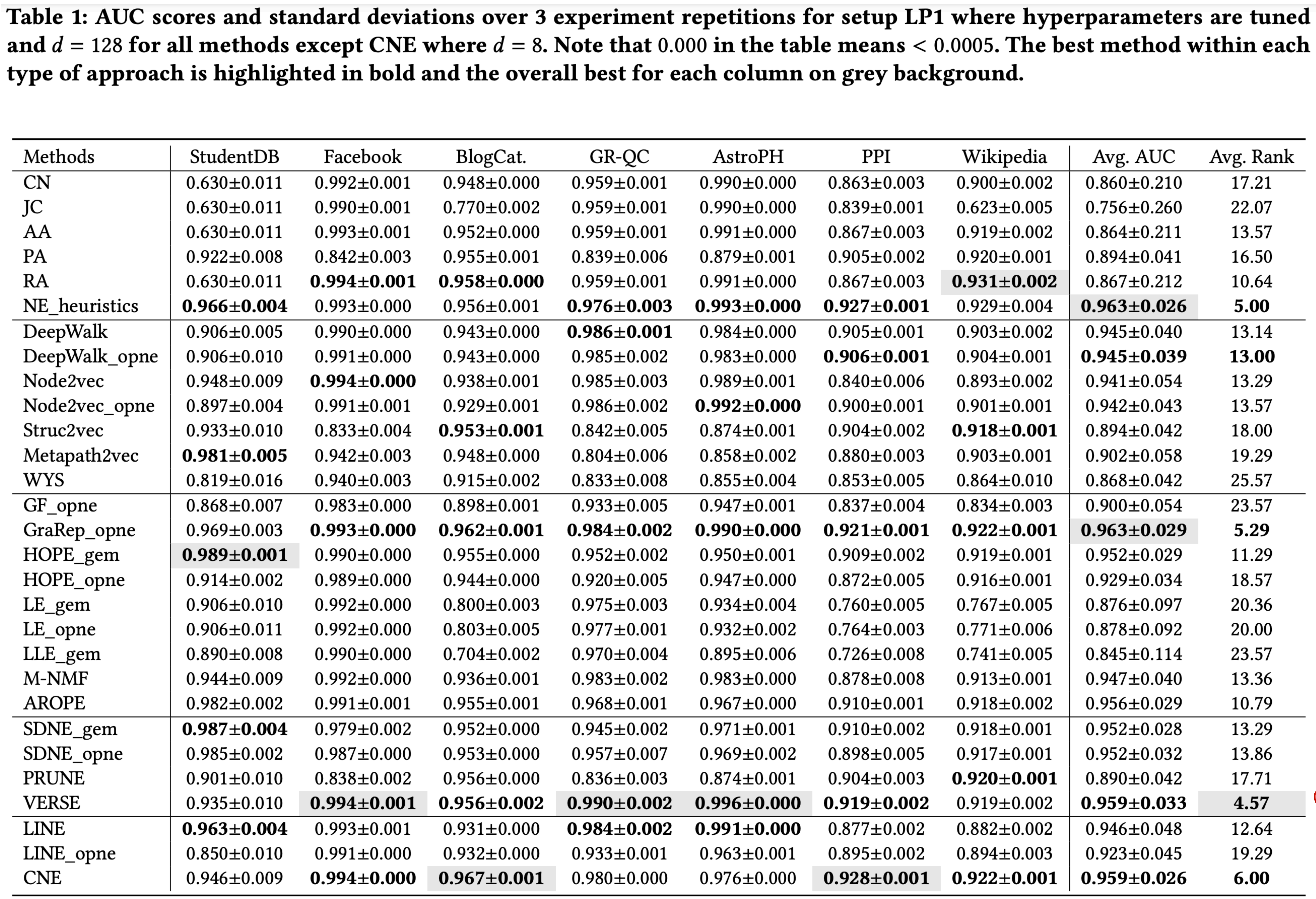


Table 1: AUC scores and standard deviations over 3 experiment repetitions for setup LP1 where hyperparameters are tuned and $d = 128$ for all methods except CNE where $d = 8$. Note that 0.000 in the table means < 0.0005. The best method within each type of approach is highlighted in bold and the overall best for each column on grey background.

Methods	StudentDB	Facebook	BlogCat.	GR-QC	AstroPH	PPI	Wikipedia	Avg. AUC	Avg. Rank
CN	0.630±0.011	0.992±0.001	0.948±0.000	0.959±0.001	0.990±0.000	0.863±0.003	0.900±0.002	0.860±0.210	17.21
JC	0.630±0.011	0.990±0.001	0.770±0.002	0.959±0.001	0.990±0.000	0.839±0.001	0.623±0.005	0.756±0.260	22.07
AA	0.630±0.011	0.993±0.001	0.952±0.000	0.959±0.001	0.991±0.000	0.867±0.003	0.919±0.002	0.864±0.211	13.57
PA	0.922±0.008	0.842±0.003	0.955±0.001	0.839±0.006	0.879±0.001	0.905±0.002	0.920±0.001	0.894±0.041	16.50
RA	0.630±0.011	**0.994±0.001**	**0.958±0.000**	0.959±0.001	0.991±0.000	0.867±0.003	**0.931±0.002**	0.867±0.212	10.64
NE_heuristics	**0.966±0.004**	0.993±0.000	0.956±0.001	**0.976±0.003**	**0.993±0.000**	**0.927±0.001**	0.929±0.004	**0.963±0.026**	**5.00**
DeepWalk	0.906±0.005	0.990±0.000	0.943±0.000	**0.986±0.001**	0.984±0.000	0.905±0.001	0.903±0.002	0.945±0.040	13.14
DeepWalk_opne	0.906±0.010	0.991±0.000	0.943±0.000	0.985±0.002	0.983±0.000	**0.906±0.001**	0.904±0.001	**0.945±0.039**	**13.00**
Node2vec	0.948±0.009	**0.994±0.000**	0.938±0.001	0.985±0.003	0.989±0.001	0.840±0.006	0.893±0.002	0.941±0.054	13.29
Node2vec_opne	0.897±0.004	0.991±0.001	0.929±0.001	0.986±0.002	**0.992±0.000**	0.900±0.001	0.901±0.001	0.942±0.043	13.57
Struc2vec	0.933±0.010	0.833±0.004	**0.953±0.001**	0.842±0.005	0.874±0.001	0.904±0.002	**0.918±0.001**	0.894±0.042	18.00
Metapath2vec	**0.981±0.005**	0.942±0.003	0.948±0.000	0.804±0.006	0.858±0.002	0.880±0.003	0.903±0.001	0.902±0.058	19.29
WYS	0.819±0.016	0.940±0.003	0.915±0.002	0.833±0.008	0.855±0.004	0.853±0.005	0.864±0.010	0.868±0.042	25.57
GF_opne	0.868±0.007	0.983±0.000	0.898±0.001	0.933±0.005	0.947±0.001	0.837±0.004	0.834±0.003	0.900±0.054	23.57
GraRep_opne	0.969±0.003	**0.993±0.000**	**0.962±0.001**	**0.984±0.002**	**0.990±0.000**	**0.921±0.001**	**0.922±0.001**	**0.963±0.029**	**5.29**
HOPE_gem	**0.989±0.001**	0.990±0.000	0.955±0.000	0.952±0.002	0.950±0.001	0.909±0.002	0.919±0.001	0.952±0.029	11.29
HOPE_opne	0.914±0.002	0.989±0.000	0.944±0.000	0.920±0.005	0.947±0.000	0.872±0.005	0.916±0.001	0.929±0.034	18.57
LE_gem	0.906±0.010	0.992±0.000	0.800±0.003	0.975±0.003	0.934±0.004	0.760±0.005	0.767±0.005	0.876±0.097	20.36
LE_opne	0.906±0.011	0.992±0.000	0.803±0.005	0.977±0.001	0.932±0.002	0.764±0.003	0.771±0.006	0.878±0.092	20.00
LLE_gem	0.890±0.008	0.990±0.000	0.704±0.002	0.970±0.004	0.895±0.006	0.726±0.008	0.741±0.005	0.845±0.114	23.57
M-NMF	0.944±0.009	0.992±0.000	0.936±0.001	0.983±0.002	0.983±0.000	0.878±0.008	0.913±0.001	0.947±0.040	13.36
AROPE	0.982±0.002	0.991±0.001	0.955±0.001	0.968±0.001	0.967±0.000	0.910±0.001	0.918±0.002	0.956±0.029	10.79
SDNE_gem	**0.987±0.004**	0.979±0.002	0.952±0.000	0.945±0.002	0.971±0.001	0.910±0.002	0.918±0.001	0.952±0.028	13.29
SDNE_opne	0.985±0.002	0.987±0.000	0.953±0.000	0.957±0.007	0.969±0.002	0.898±0.005	0.917±0.001	0.952±0.032	13.86
PRUNE	0.901±0.010	0.838±0.002	0.956±0.000	0.836±0.003	0.874±0.001	0.904±0.003	**0.920±0.001**	0.890±0.042	17.71
VERSE	0.935±0.010	**0.994±0.001**	**0.956±0.002**	**0.990±0.002**	**0.996±0.000**	**0.919±0.002**	0.919±0.002	**0.959±0.033**	**4.57**
LINE	**0.963±0.004**	0.993±0.001	0.931±0.000	**0.984±0.002**	**0.991±0.000**	0.877±0.002	0.882±0.002	0.946±0.048	12.64
LINE_opne	0.850±0.010	0.991±0.000	0.932±0.000	0.933±0.001	0.963±0.001	0.895±0.002	0.894±0.003	0.923±0.045	19.29
CNE	0.946±0.009	**0.994±0.000**	**0.967±0.001**	0.980±0.000	0.976±0.000	**0.928±0.001**	**0.922±0.001**	**0.959±0.026**	**6.00**

Link prediction: VERSE is the best-ranking method in an independent study

(Mara *et al.*, 2020)