The Expressive Power of Graph Neural Networks (PART 2)

1/30/2025

Arun Cheriakara Joseph,
Faculty Of Math

UNIVERSITY OF WATERLOO

What Is the Problem?

- **Core Issue**: GNNs rely on **local** message passing
- Often **can't** capture:
  - They can fail to distinguish **nodes or subgraphs** that look similar in small neighborhoods (like counting cycles, differentiating long-range distances, or separating certain regular graphs).

Why Is It Important?

- Many real-world tasks (link prediction, graph classification, node role labeling) require **long-range structural insight**.
- If GNNs can't handle distances, they may fail in domains like **social networks**, **chemistry**, or **knowledge graphs**.

Why Don't Previous Methods Work?

- **Standard MP-GNN:**
  - Essentially a **1-WL** (Weisfeiler–Lehman) **isomorphism test** → limited in distinguishing certain symmetric graphs.
- **Local Aggregation Only**:
  - **Shallow** GNN layers fail if structural differences appear **beyond** a few hops.
- **No Positional/Distance** info:
  - GNNs without explicit distance or identity labels can't break symmetrical patterns.

What Is the Proposed Solution?

- **Randomized Matrix Factorization** (Srinivasan & Ribeiro 2020, Dwivedi et al. 2020)
- **Deterministic Distance Attributes** (Li et al. 2020e)
- **ID-GNN** (You et al. 2021)

UNIVERSITY OF WATERLOO

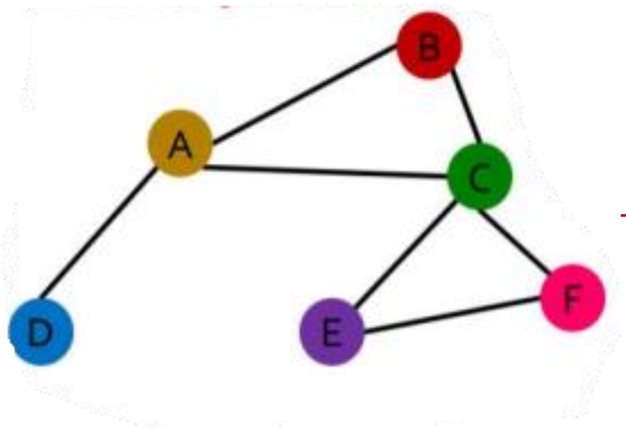
RANDOMIZED MATRIX FACTORIZATION

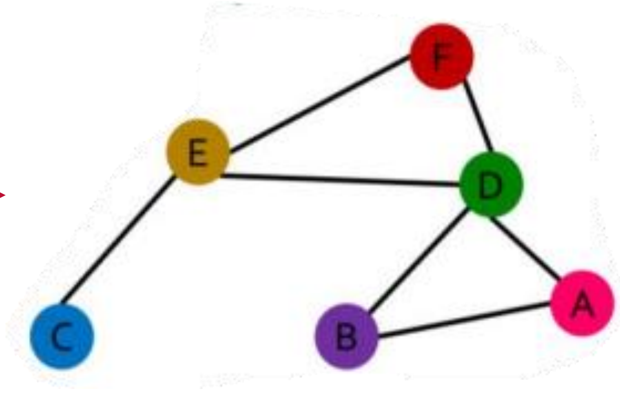
Goal: Understand how randomized node embeddings preserve permutation invariance & help GNNs

SRINIVASAN & RIBEIRO (2020) AND DWIVEDI ET AL. (2020)

GNNs & Permutation Invariance

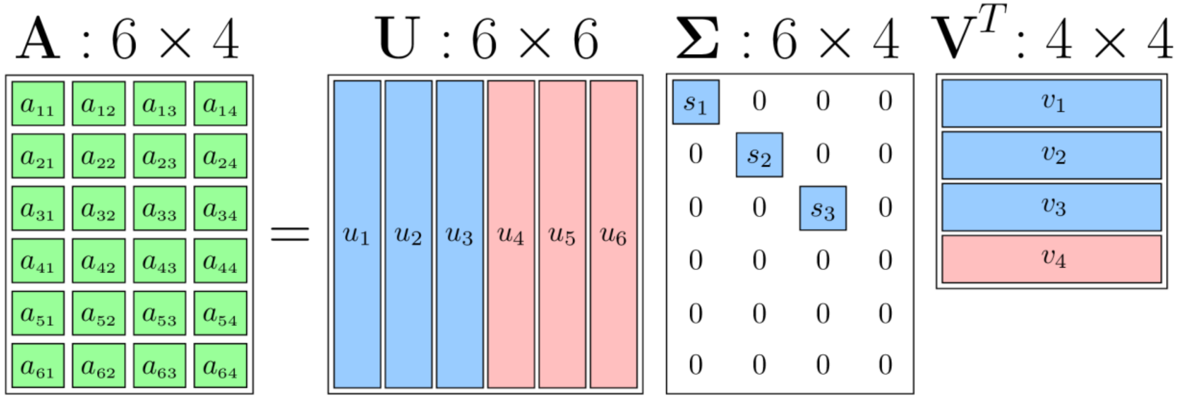
- GNNs encode graph structure in permutation-invariant ways
- Relabeling nodes shouldn’t change the output
- Traditional GNNs often lack explicit positional or structural node info

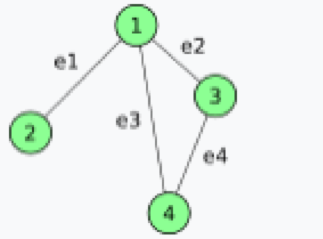




Matrix Factorization Basics

- Adjacency matrix A or Laplacian L
- Singular value decomposition (SVD): $A = U\Sigma U^T$
- Eigen-decomposition $L = U\Lambda U^T$
- Each row of $U \rightarrow$ Node embedding



Undirected graph	Incidence matrix	Laplacian matrix
	$\begin{pmatrix} 1 & 1 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & -1 \end{pmatrix}$	$\begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}$

UNIVERSITY OF WATERLOO

Non-Uniqueness & Random Perturbations

- Non-uniqueness: SVD/eigen decompositions can differ by sign flips, column order, etc.
- Random sign flips or noise → unify different valid decompositions
- Preserves permutation invariance in expectation

Srinivasan & Ribeiro (2020) – Approach

- Proposed concept: random factorization → node embeddings
- Didn't do explicit SVD in practice
- Used random Gaussian matrices + graph propagation
- e.g., for the two hops: $Z_G = \psi(\hat{A}\psi(\hat{A}Z_{G1}) + Z_{G2})$,
- where:
  - Z_{G1}, Z_{G2} = Gaussian random matrices
  - ψ= MLP
  - $\hat{A}$ = adjacency matrix
  - Rows of Z_G = final node embeddings

Dwivedi et al. (2020) Approach

- Explicit eigen-decomposition of the normalized Laplacian
- $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, then $L = U\Lambda U^T$
- where:
  - $\hat{A}$ = adjacency matrix
  - D = diagonal degree matrix
  - Λ is a diagonal matrix of eigenvalues
  - U = corresponding eigenvectors
- $Z_{LE} = U\ \Gamma^T$ with random ±1 sign flips

Permutation Invariance in Expectation

- Key claim: Random sign flips preserve permutation invariance
- If the graph is relabeled ⇒ factorization permutes the same way
- Lemma 5.3 & Theorem 5.8
  - Random sign flips preserve **permutation invariance** *in expectation*
  - If **eigenvalues** of L are distinct
  - Ensures consistency under **node relabeling**

Takeaways

- Factorizing A or L → “positional” node embeddings
- **Random perturbations** = crucial to preserve invariance & inductive power
- Empirical **trade-offs**: not always top performance vs. alternatives like distance encoding

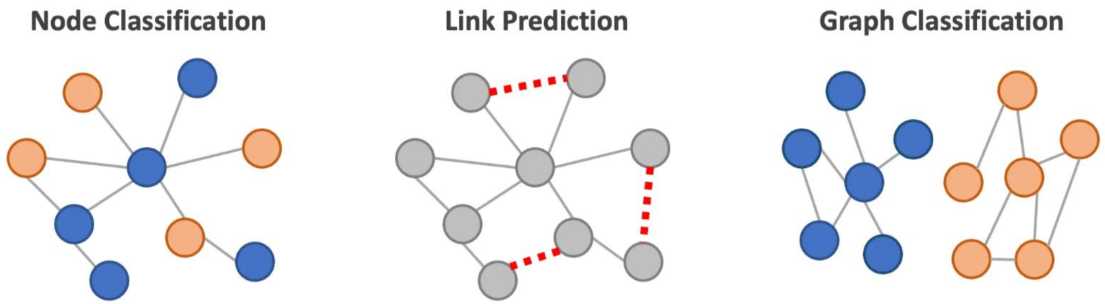
UNIVERSITY OF WATERLOO

Injecting Deterministic Distance Attributes

- **Problem**: MP-GNNs struggle to measure **long-range distances**, **count cycles**, etc.
- **Solution**: Inject **deterministic** distance features into nodes/edges
- **Result**: Boost the **expressive power** of MP-GNN

Designing Deterministic Distance Attributes

- **Task-Specific** Distance Info
  - Node classification ($|S| = 1$): distance from node to itself
  - Link prediction ($|S| = 2$): distance between two end nodes
  - Graph-level ($S = V(G)$) distances among all node pairs



UNIVERSITY OF WATERLOO

Applications

- **SEAL** (Zhang & Chen, 2018b):
  - Extract **enclosing subgraph**
  - Annotate each node w/ **shortest-path distance** to end-nodes
- **Chen et al. (2019a), Maziarka et al. (2020a)**:
  - Use Shortest Path Distance (SPDs) as **edge attributes**
- **You et al. (2021)**:
  - Mark target node as 1, others as 0 in node classification

Comparing Deterministic vs. Random Attributes

- **Deterministic** Pros:
  - Less **noise**, faster convergence
  - Often better **generalization** in practice
- **Deterministic** Cons:
  - May **lack universal approximation**
  - Must be **recomputed** for each query S
- **Random** (from earlier) can be universal in a **probabilistic** sense

DISTANCE ENCODING

Goal: **Attach extra node attributes** to a GNN which captures distance between nodes

LI ET AL., 2020E

Overview of Distance Encoding

- **Motivation**: Enhance MP-GNN with **explicit** distance information
- **Key Idea**: Define $\zeta(u|S)$ = “distance encoding” for node u w.r.t. subset S
- **Goal**: Make GNN more expressive (e.g., differentiate structurally similar nodes)

Definition of Distance Encoding

- **Equation 5.14**:

$$\zeta(u \mid S) = \sum_{v \in S} MLP(\zeta(u \mid v))$$

- **ζ(u|v)** = pairwise distance descriptor from node u to v
- **Interpretation**: Summation of MLP outputs, one per v∈S

Defining ζ(u|v)

- **ζ(u|v)** (*pairwise* distance descriptor between u and v) can be computed in multiple ways:
  - **Shortest-path distance** (SPD)
  - **Random-walk** or **PageRank** distances
  - **Heat-kernel** or other spectral distances
- **Example**: $\ell_{uv} = (1, (W)_{uv}^2, (W)_{uv}^3, \ldots), Where\ W = AD^{-1}$
  - Then $g(\ell_{uv})$ picks out a distance measure
  - $\zeta(u|v) = g(\ell_{uv})$

DE-GNN: Using Distance Encoding as Node Attributes

- **Concatenate** $\zeta(v \mid S)$ with node features:

$$\tilde{X}_v = X_v \oplus \zeta(v \mid S)$$

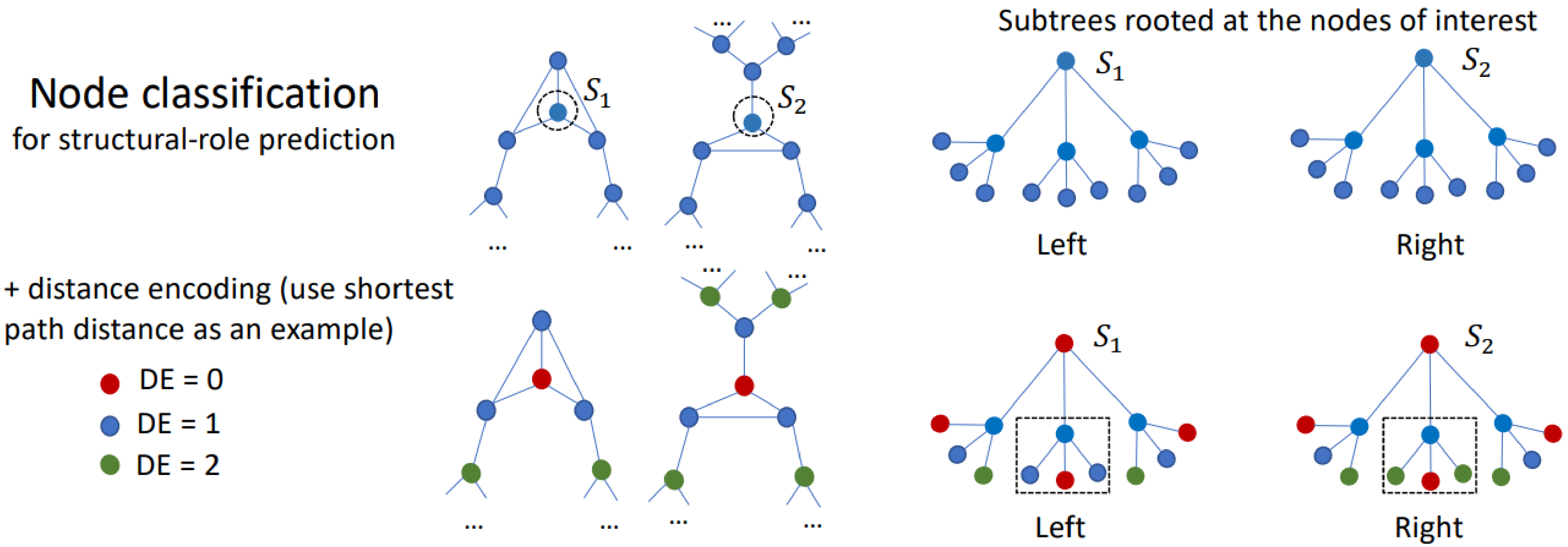
- **Feed** $\{\tilde{X}_v\}$into an MP-GNN
- Result: A model called **DE-GNN**

Expressive Power: Lemma 5.4 & Theorem 5.9

- **Lemma 5.4**: Permutation-invariance still holds for **isomorphic** graphs
- **Theorem 5.9**: DE-GNN can distinguish certain **regular graphs** that MP-GNN cannot
- **Implication**: **Stronger** than standard MP-GNN on tricky graphs

Node Classification

- S1 vs. S2 in the figure—MP-GNN might confuse them, but DE-GNN can tell them apart.



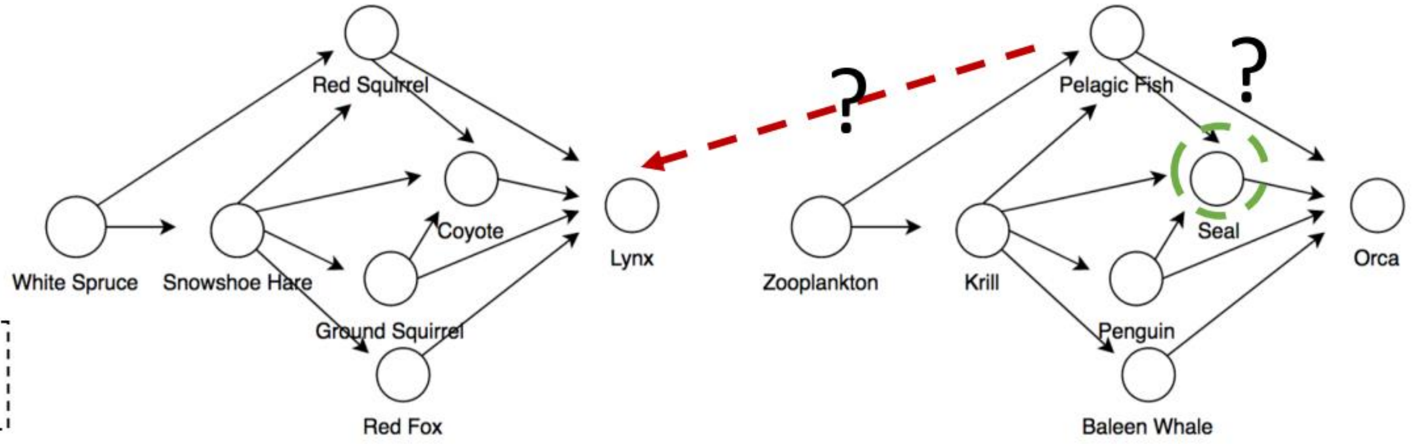
Link Prediction

- Without node identities, these pairs look **isomorphic**

Link prediction

+ distance encoding (use shortest path distance as an example)

$$\zeta(Seal|\{Orca, Pelagic\ Fish\}) = \{1,1\}$$
$$\zeta(Seal|\{Lynx, Pelagic\ Fish\}) = \{1,\infty\}$$



Caveats & Limitations

- Not universally expressive (some distance-regular graphs remain indistinguishable)
- Additional **computational cost** for distance metrics

IDENTITY-AWARE GNN

Goal: Simplify distance encoding for **single-node** tasks

(YOU ET AL, 2021)

Intro to Identity-aware GNN (ID-GNN)

- **Motivation**: Simplify distance encoding for **single-node** tasks
- **Key Idea**: Attach a **binary attribute** $\zeta ID(u \mid \{v\})$

$$\zeta ID(u \mid \{v\}) = \begin{cases} 1 \; if \; u = v \\ 0 \; otherwise \end{cases}$$

- **Focus**: Node classification (where |S|=1)

UNIVERSITY OF WATERLOO

ID-GNN vs. DE-GNN for Node Classification

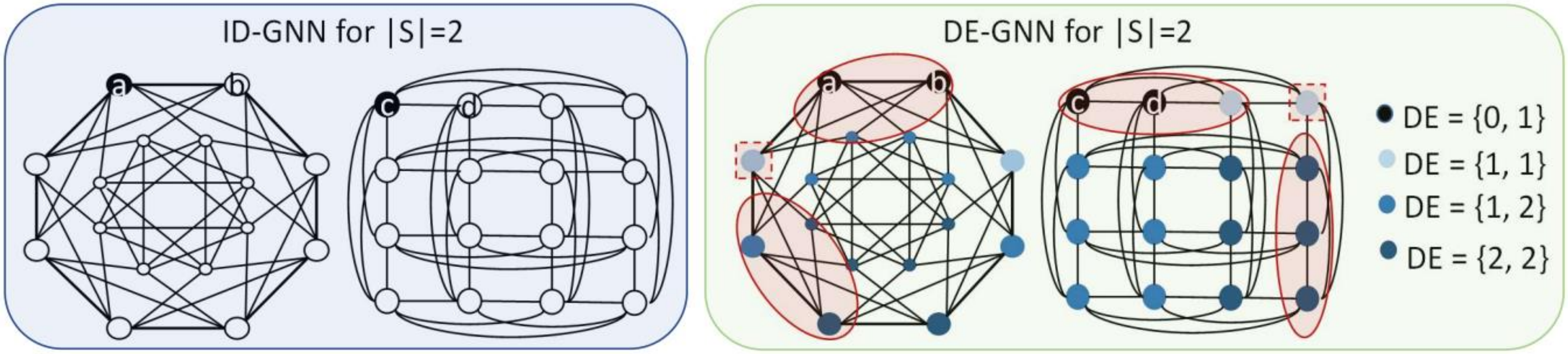
- When |S|=1, ID-GNN **matches** DE-GNN's power
- The **1** bit acts like distance = 0, everything else distance = ∞
- **Same** representation power, sometimes more layers needed

Theorem 5.10 – Layer Complexity

- **Statement**: "If DE-GNN distinguishes two examples in L layers, ID-GNN does so in **at most** 2L layers."
- **Reason**: 1st L layers to spread identity info, 2nd L layers to gather it back
- **Implication**: ID-GNN is **less layer-efficient**, but equally strong in principle

ID-GNN vs DE-GNN

- Each graph has a pair of target nodes (like {a,b} or {c,d}).
- **ID-GNN**: not designed for multi-node identity → struggles or needs extra passes
- **DE-GNN**: can label each node's distance to both targets at once → easier distinction



UNIVERSITY OF WATERLOO

Wrap-Up on ID-GNN

- **Power**: Matches DE-GNN for single-node tasks
- Limitations:
  - Potentially **2**× deeper GNN needed
  - Doesn't natively handle $|S| \geq 2$
- **Practical**: Simpler than distance calculations, less overhead for single-target tasks

What Interesting Research Questions Remain?

- **Explored**
  - **Random Factorization**: Adds global "positional" info, might need sign flips
  - **Deterministic Distances**: Often strong in practice, e.g. SEAL, DE-GNN
  - **ID-GNN**: Handy for single-node tasks, needs more layers or separate runs for $|S| \geq 2$
- **Future Research**:
  - **Efficiency** of distance computations or large matrix factorizations
  - **Hybrid** embeddings: Combine random & deterministic?

UNIVERSITY OF WATERLOO

Thank You!