

A PROOFS

Proposition 2.4 SP4LP does not suffer from the automorphic node problem.

Proof. According to Proposition 1.9, a model M suffers from the *automorphic node problem* if, for any graph $G = (V, E, X^0)$, for any pairs of links $(u, v), (u', v') \in V \times V$, for any $F \in M$ it holds that:

$$(u, v) \not\sim (u', v'), \quad u \simeq u', \quad v \simeq v', \quad \text{and} \quad F((u, v), G) \neq F((u', v'), G).$$

In order to prove that SP4LP does not suffer from the automorphic node problem, it suffices to provide an example of a graph $G = (V, E, X^0)$ and node pairs $(u, v), (u', v') \in V \times V$ such that:

$$(u, v) \not\sim (u', v'), \quad u \simeq u', \quad v \simeq v', \quad \text{and} \quad \text{SP4LP}((u, v), G) \neq \text{SP4LP}((u', v'), G).$$

Such an example is provided in Figure 1: the shortest path from v to u' consists of v , an orange node, and u' , whereas the shortest path from v to u includes v , an orange node, a yellow node, another orange node, and finally u . Thus, simply using a summation as function ϕ , leads to distinct representations for links (v, u) and (v, u') . \square

Theorem 2.5 SP4LP is strictly more expressive than Pure GNN, NCN, BUDDY, NBFnet and Neo-GNN.

Proof. We proceed by prove each comparison separately.

- SP4LP is strictly more expressive than Pure GNN.

We prove this by noting that SP4LP architecture (Equation 8) generalizes that of pure GNNs, meaning that SP4LP can simulate any pure GNN by simply ignoring the shortest path information. Thus, for any pair of non-automorphic links that a specific pure GNN can distinguish, there exists a configuration of SP4LP that distinguishes them as well. We can conclude that SP4LP is strictly more expressive than pure GNNs considering as example of pair of links indistinguishable by GNNs but distinguishable by SP4LP the one provided in the proof of Proposition 2.4.

- SP4LP is strictly more expressive than NCN.

We prove this in two steps: (1) When two links share no common neighbors, NCN on them reduces to a pure GNN. As we have already proved that SP4LP is strictly more expressive than pure GNNs, it follows that SP4LP is also strictly more expressive than NCN in this case. (2) When the links have common neighbors, setting AGG as summation, ρ as the Hadamard product between the endpoint representations and the concatenation with the result of the aggregation, and ϕ as the identity function, SP4LP reduces exactly to NCN. Therefore, if NCN can distinguish two links under some configuration, SP4LP can as well. By definition 1.10, this implies that SP4LP is more expressive than NCN. We can conclude that SP4LP is strictly more expressive than NCN considering the example in Figure 4. The graph is regular and thus all nodes receive the same embedding from a GNN. Consider nodes u and v : $N(u) \cap N(v) = N(u) \cap N(v') = \emptyset$. In this case, NCN reduces to a pure GNN and is thus unable to distinguish the links (u, v) and (u', v') . Now, let $\mathbf{x} \in \mathbb{R}^d$ be the representation assigned to every node by a GNN. Then, the representation of the shortest path $\mathcal{P}_G^*(u, v)$ is simply $(\mathbf{x}, \mathbf{x}, \mathbf{x})$, while $\mathcal{P}_G^*(u', v') = (\mathbf{x}, \mathbf{x}, \mathbf{x}, \mathbf{x})$. Even using a simple sum as aggregation function, SP4LP successfully distinguishes between the two links.

- SP4LP is strictly more expressive than Neo-GNN and BUDDY.

We have already shown that SP4LP is strictly more expressive than NCN. In Theorem 2 of the NCN paper Wang et al. (2024), it has been proved that NCN is more expressive than both Neo-GNN and BUDDY. It follows that SP4LP is strictly more expressive than Neo-GNN and BUDDY as well.

- SP4LP is strictly more expressive than NBFNet.

We prove that NBFNet is as expressive as a pure GNN. Therefore, since we have already proven that SP4LP is strictly more expressive than pure GNNs, it immediately follows that SP4LP is also strictly more expressive than NBFNet.

To complete the argument, we prove that NBFNet is as expressive as a pure GNN. First of all we report the formulation of NBFNet for simple undirected graph. Given a graph $G = (V, E, \mathbf{X}^0)$, NBFNet assigns a representation $\mathbf{x}(u, v)$ to each edge $(u, v) \in E$. The iterative update rule follows a message-passing scheme:

$$\begin{aligned} \mathbf{x}_{(u,v)}^{(0)} &= \text{INDICATOR}(\mathbf{x}_u^0, \mathbf{x}_v^0), \\ \mathbf{x}_{(u,v)}^{(l)} &= \text{AGGREGATE} \left(\left\{ \text{MESSAGE}(h_{(i,j)}^{(l-1)}) \mid (i, j) \in N(u, v) \right\} \cup \{h_{(u,v)}^{(0)}\} \right) \end{aligned} \quad (9)$$

where INDICATOR assigns an initial representation based on the nodes $u, v \in V$ and $N(u, v)$ is the set of edges incident to (u, v) .

We prove that, at any layer l , the representations of the two links produced by a pure GNN are equal if and only if also the ones produced by NBFNet are equal, i.e.,

$$\mathbf{x}_{(u,v)}^{\text{GNN}^l} = \mathbf{x}_{(i,j)}^{\text{GNN}^l} \Leftrightarrow \mathbf{x}_{(u,v)}^{\text{NBF}^l} = \mathbf{x}_{(i,j)}^{\text{NBF}^l} \quad \forall l \quad (10)$$

where $\mathbf{x}_{(u,v)}^{\text{NBF}^l}$ and $\mathbf{x}_{(i,j)}^{\text{NBF}^l}$ are calculated via Equation 9, while $\mathbf{x}_{(u,v)}^{\text{GNN}^l}$ and $\mathbf{x}_{(i,j)}^{\text{GNN}^l}$ are calculated following the standard message passing scheme reported below:

$$\begin{aligned} \mathbf{x}_v^{\text{GNN}^0} &= \mathbf{x}_v^0, \\ \mathbf{x}_v^{\text{GNN}^l} &= \text{COMB} \left(\mathbf{x}_v^{\text{GNN}^{l-1}}, \text{AGG} \left(\{ \mathbf{x}_u^{\text{GNN}^{l-1}} \mid u \in N(v) \} \right) \right) \\ \mathbf{x}_{(u,v)}^{\text{GNN}^l} &= g(\mathbf{x}_u^{\text{GNN}^l}, \mathbf{x}_v^{\text{GNN}^l}) \end{aligned} \quad (11)$$

Let the functions INDICATOR, AGGREGATE and MESSAGE of Equation 9, as well as the functions COMB, AGG and g be injective. We prove Equation 10 by induction on the number of layer l .

Base Case: $l = 0$

$$\begin{aligned} \mathbf{x}_{(u,v)}^{\text{GNN}^0} &= \mathbf{x}_{(i,j)}^{\text{GNN}^0} \xLeftrightarrow{\text{11}} g(\mathbf{x}_u^0, \mathbf{x}_v^0) = g(\mathbf{x}_i^0, \mathbf{x}_j^0) \xLeftrightarrow{\text{inj}} (\mathbf{x}_u^0, \mathbf{x}_v^0) = (\mathbf{x}_i^0, \mathbf{x}_j^0) \xLeftrightarrow{\text{inj}} \\ &\xLeftrightarrow{\text{inj}} \text{INDICATOR}(\mathbf{x}_u^0, \mathbf{x}_v^0) = \text{INDICATOR}(\mathbf{x}_i^0, \mathbf{x}_j^0) \xLeftrightarrow{\text{9}} \mathbf{x}_{(u,v)}^{\text{NBF}^0} = \mathbf{x}_{(i,j)}^{\text{NBF}^0} \end{aligned}$$

Inductive Step We assume Equation 10 holds for $l - 1$ and prove it holds for l .

In particular, we want to prove

$$\mathbf{x}_{(u,v)}^{\text{GNN}^l} = \mathbf{x}_{(i,j)}^{\text{GNN}^l} \Leftrightarrow \mathbf{x}_{(u,v)}^{\text{NBF}^l} = \mathbf{x}_{(i,j)}^{\text{NBF}^l} \quad (12)$$

using the inductive hypothesis

$$\mathbf{x}_{(u,v)}^{\text{GNN}^{l-1}} = \mathbf{x}_{(i,j)}^{\text{GNN}^{l-1}} \Leftrightarrow \mathbf{x}_{(u,v)}^{\text{NBF}^{l-1}} = \mathbf{x}_{(i,j)}^{\text{NBF}^{l-1}} \quad (13)$$

Applying Equation 11 to the left-hand side of Equation 12 we get

$$\begin{aligned} \mathbf{x}_{(u,v)}^{\text{GNN}^l} &= \mathbf{x}_{(i,j)}^{\text{GNN}^l} \\ &\xLeftrightarrow{\text{11}} \\ &g(\text{COMB}(\mathbf{x}_u^{\text{GNN}^{l-1}}, \text{AGG}(\{ \mathbf{x}_x^{\text{GNN}^{l-1}} \mid x \in N(u) \})), \text{COMB}(\mathbf{x}_v^{\text{GNN}^{l-1}}, \text{AGG}(\{ \mathbf{x}_y^{\text{GNN}^{l-1}} \mid y \in N(u) \}))) \\ &= \\ &g(\text{COMB}(\mathbf{x}_i^{\text{GNN}^{l-1}}, \text{AGG}(\{ \mathbf{x}_m^{\text{GNN}^{l-1}} \mid m \in N(i) \})), \text{COMB}(\mathbf{x}_j^{\text{GNN}^{l-1}}, \text{AGG}(\{ \mathbf{x}_n^{\text{GNN}^{l-1}} \mid n \in N(j) \}))). \end{aligned}$$

Given the injectivity of g , COMB and AGG, this is equivalent to

$$\begin{aligned}
& \mathbf{x}_u^{\text{GNN}^{l-1}} = \mathbf{x}_i^{\text{GNN}^{l-1}} \wedge \{\mathbf{x}_x^{\text{GNN}^{l-1}} \mid x \in N(u)\} = \{\mathbf{x}_m^{\text{GNN}^{l-1}} \mid m \in N(i)\} \wedge \\
& \wedge \mathbf{x}_v^{\text{GNN}^{l-1}} = \mathbf{x}_j^{\text{GNN}^{l-1}} \wedge \{\mathbf{x}_y^{\text{GNN}^{l-1}} \mid y \in N(v)\} = \{\mathbf{x}_n^{\text{GNN}^{l-1}} \mid n \in N(j)\} \\
& \iff \\
& \{(\mathbf{x}_u^{\text{GNN}^{l-1}}, \mathbf{x}_x^{\text{GNN}^{l-1}}) \mid x \in N(u)\} = \{(\mathbf{x}_i^{\text{GNN}^{l-1}}, \mathbf{x}_m^{\text{GNN}^{l-1}}) \mid m \in N(i)\} \\
& \wedge \\
& \{(\mathbf{x}_v^{\text{GNN}^{l-1}}, \mathbf{x}_y^{\text{GNN}^{l-1}}) \mid y \in N(v)\} = \{(\mathbf{x}_j^{\text{GNN}^{l-1}}, \mathbf{x}_n^{\text{GNN}^{l-1}}) \mid n \in N(j)\} \\
& \iff \\
& \{(\mathbf{x}_u^{\text{GNN}^{l-1}}, \mathbf{x}_x^{\text{GNN}^{l-1}}) \mid x \in N(u)\} \cup \{(\mathbf{x}_v^{\text{GNN}^{l-1}}, \mathbf{x}_y^{\text{GNN}^{l-1}}) \mid y \in N(v)\} \\
& = \\
& \{(\mathbf{x}_i^{\text{GNN}^{l-1}}, \mathbf{x}_m^{\text{GNN}^{l-1}}) \mid m \in N(i)\} \cup \{(\mathbf{x}_j^{\text{GNN}^{l-1}}, \mathbf{x}_n^{\text{GNN}^{l-1}}) \mid n \in N(j)\}.
\end{aligned}$$

By Definition of $N(u, v)$ (Equation 9), this is equivalent to

$$\begin{aligned}
& \{(\mathbf{x}_w^{\text{GNN}^{l-1}}, \mathbf{x}_t^{\text{GNN}^{l-1}}) \mid (w, t) \in N(u, v)\} = \{(\mathbf{x}_a^{\text{GNN}^{l-1}}, \mathbf{x}_b^{\text{GNN}^{l-1}}) \mid (a, b) \in N(i, j)\} \\
& \xleftrightarrow{\text{inj}} \\
& \{g(\mathbf{x}_w^{\text{GNN}^{l-1}}, \mathbf{x}_t^{\text{GNN}^{l-1}}) \mid (w, t) \in N(u, v)\} = \{g(\mathbf{x}_a^{\text{GNN}^{l-1}}, \mathbf{x}_b^{\text{GNN}^{l-1}}) \mid (a, b) \in N(i, j)\} \\
& \xleftrightarrow{(11)} \\
& \{\mathbf{x}_{(w,t)}^{\text{GNN}^{l-1}} \mid (w, t) \in N(u, v)\} = \{\mathbf{x}_{(a,b)}^{\text{GNN}^{l-1}} \mid (a, b) \in N(i, j)\} \\
& \xleftrightarrow{\text{IND. HYP. (13)}} \\
& \{\mathbf{x}_{(w,t)}^{\text{NBF}^{l-1}} \mid (w, t) \in N(u, v)\} = \{\mathbf{x}_{(a,b)}^{\text{NBF}^{l-1}} \mid (a, b) \in N(i, j)\}.
\end{aligned}$$

Using the hypotheses of injective AGG and MESSAGE, this is equivalent to:

$$\begin{aligned}
& \text{AGG}(\{\text{MESSAGE}(\mathbf{x}_{(w,t)}^{\text{NBF}^{l-1}}) \mid (w, t) \in N(u, v)\}) = \text{AGG}(\{\text{MESSAGE}(\mathbf{x}_{(a,b)}^{\text{NBF}^{l-1}}) \mid (a, b) \in N(i, j)\}) \\
& \xleftrightarrow{(9)} \\
& \mathbf{x}_{(u,v)}^{\text{NBF}^l} = \mathbf{x}_{(i,j)}^{\text{NBF}^l} \tag{14}
\end{aligned}$$

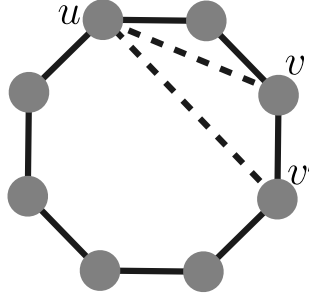
which complete the proof. \square

B ADDITIONAL RESULTS

We complement the main results of Section 4 with additional tables reporting the standard deviation computed over five different random seeds, to better assess the stability of each method.

Real-world Datasets: Results with Standard Deviations Table 4 and Table 5 expand the main results in Table 2 by including both the mean and standard deviation of MRR and Hits@ K across runs.

Ablation Study: Results with Standard Deviations Similarly, Table 6 complements the ablation results in Table 3 by reporting mean and standard deviation for Cora, Citeseer, and Pubmed.

Figure 4: Links (u, v) , (u, v') are not distinguished by NCN while are distinguished by SP4LP.

Models	Cora MRR	Citeseer MRR	Pubmed MRR	Ogbl-ddi MRR	Ogbl-collab MRR	Ogbl-ppa MRR	Ogbl-Citation2 MRR
Node2Vec	14.47 (± 0.60)	21.17 (± 1.01)	3.94 (± 0.24)	11.14 (± 0.95)	4.68 (± 0.08)	18.33 (± 0.10)	14.67 (± 0.18)
MF	6.20 (± 1.42)	7.80 (± 0.79)	4.46 (± 0.32)	13.99 (± 0.47)	4.89 (± 0.25)	22.47 (± 1.53)	8.72 (± 2.60)
MLP	13.52 (± 0.65)	22.62 (± 0.55)	6.41 (± 0.25)	N/A	5.37 (± 0.14)	0.98 (± 0.00)	16.32 (± 0.07)
GCN	16.61 (± 0.30)	21.09 (± 0.88)	7.13 (± 0.27)	13.46 (± 0.34)	6.09 (± 0.38)	26.94 (± 0.48)	19.98 (± 0.35)
GAT	13.84 (± 0.68)	19.58 (± 0.84)	4.95 (± 0.14)	12.92 (± 0.39)	4.18 (± 0.33)	OOM	OOM
SAGE	14.74 (± 0.69)	21.09 (± 1.15)	9.40 (± 0.70)	12.60 (± 0.72)	5.53 (± 0.50)	27.27 (± 0.30)	22.05 (± 0.12)
GAE	18.32 (± 0.41)	25.25 (± 0.82)	5.27 (± 0.25)	3.49 (± 1.73)	OOM	OOM	OOM
SEAL	10.67 (± 3.46)	13.16 (± 1.66)	5.88 (± 0.53)	9.99 (± 0.90)	6.43 (± 0.32)	29.71 (± 0.71)	20.60 (± 1.28)
BUDDY	13.71 (± 0.59)	22.84 (± 0.36)	7.56 (± 0.18)	12.43 (± 0.50)	5.67 (± 0.36)	27.70 (± 0.33)	19.17 (± 0.10)
Neo-GNN	13.95 (± 0.39)	17.34 (± 0.84)	7.74 (± 0.30)	10.86 (± 2.16)	5.23 (± 0.90)	21.68 (± 1.14)	16.12 (± 0.25)
NCN	14.66 (± 0.95)	28.65 (± 1.21)	5.84 (± 0.22)	12.86 (± 0.78)	5.09 (± 0.38)	35.06 (± 0.26)	23.35 (± 0.28)
NCNC	14.98 (± 1.00)	24.10 (± 0.65)	8.58 (± 0.59)	>24h	4.73 (± 0.86)	33.52 (± 0.26)	19.61 (± 0.54)
NBFNet	13.56 (± 0.58)	14.29 (± 0.80)	>24h	>24h	OOM	OOM	OOM
PEG	15.73 (± 0.39)	21.01 (± 0.77)	4.40 (± 0.41)	12.05 (± 1.14)	4.83 (± 0.21)	OOM	OOM
LPFORMER	16.80 (± 0.52)	26.34 (± 0.67)	9.99 (± 0.52)	13.20 (± 0.54)	7.62 (± 0.26)	40.25 (± 0.24)	24.70 (± 0.55)
SP4LP	17.27 (± 0.57)	41.08 (± 1.84)	10.87 (± 0.31)	15.00 (± 0.57)	9.46 (± 0.55)	36.45 (± 1.21)	24.91 (± 0.41)

Table 4: MRR results across all datasets, following the HeaRT evaluation setting [Li et al. \(2023\)](#). The top three results for each metric are highlighted using **first**, **second**, and **third**. OOM indicates that the model ran out of memory, while >24h denotes that the method did not complete within 24 hours.

C DATASETS STATISTICS

Table 7 summarizes the main datasets used in our link prediction experiments. Cora, Citeseer, and Pubmed are well-known citation networks frequently used as benchmarks for graph-based learning methods. These datasets are relatively small, both in the number of nodes and edges. In contrast, the datasets from the Open Graph Benchmark (OGB), namely ogbl-collab and ogb-ddi, are substantially larger and more complex, offering challenging scenarios for evaluating model scalability and performance on large-scale graphs.

For all datasets, we adopt the splits provided by the [Li et al. \(2023\)](#) setting.

D EXPERIMENTAL SETTINGS

This section outlines the experimental setup used to evaluate all models. We describe the computational resources and the hyperparameter search space. Moreover for SP4LP we include details regarding how the calculation of the shortest path is performed. Details are reported below.

Computational Resources All experiments were conducted on a workstation running Ubuntu 22.04 with an AMD Ryzen 9 7950X CPU (32 threads), 124GB of RAM, and two NVIDIA GeForce RTX 4090 GPUs (24GB each).

Hyperparameters All models are tuned using a grid search over learning rate $\in [1 \times 10^{-2}, 1 \times 10^{-3}]$, dropout $\in [0, 0.7]$, weight decay $\in [0, 10^{-4}, 10^{-7}]$, number of GNN layers $\in \{1, 2, 3\}$, hid-

Models	Cora Hits@10	Citeseer Hits@10	Pubmed Hits@10	Ogbl-ddi Hits@20	Ogbl-collab Hits@20	Ogbl-ppa Hits@20	Ogbl-Citation2 Hits@20
Node2Vec	32.77 (± 1.29)	45.82 (± 2.01)	8.51 (± 0.77)	63.63 (± 2.05)	16.84 (± 0.17)	53.42 (± 0.11)	42.68 (± 0.20)
MF	15.26 (± 3.39)	16.72 (± 1.99)	9.42 (± 0.80)	59.50 (± 1.68)	18.86 (± 0.40)	70.71 (± 4.82)	29.64 (± 7.30)
MLP	31.01 (± 1.71)	48.02 (± 1.79)	15.04 (± 0.67)	N/A	16.15 (± 0.27)	1.47 (± 0.00)	43.15 (± 0.10)
GCN	36.26 (± 1.14)	47.23 (± 1.88)	15.22 (± 0.57)	64.76 (± 1.45)	22.48 (± 0.81)	68.38 (± 0.73)	51.72 (± 0.46)
GAT	32.89 (± 1.27)	45.30 (± 1.30)	9.99 (± 0.64)	66.83 (± 2.23)	18.30 (± 1.42)	OOM	OOM
SAGE	34.65 (± 1.47)	48.75 (± 1.85)	20.54 (± 1.40)	67.19 (± 1.18)	21.26 (± 1.32)	69.49 (± 0.43)	53.13 (± 0.15)
GAE	37.95 (± 1.24)	49.65 (± 1.48)	10.50 (± 0.46)	17.81 (± 9.80)	OOM	OOM	OOM
SEAL	24.27 (± 6.74)	27.37 (± 3.20)	12.47 (± 1.23)	49.74 (± 2.39)	21.57 (± 0.38)	76.77 (± 0.94)	48.62 (± 1.93)
BUDDY	30.40 (± 1.18)	48.35 (± 1.18)	16.78 (± 0.53)	58.71 (± 1.63)	23.35 (± 0.73)	71.50 (± 0.68)	47.81 (± 0.37)
Neo-GNN	31.27 (± 0.72)	41.74 (± 1.18)	17.88 (± 0.71)	51.94 (± 10.33)	21.03 (± 3.39)	64.81 (± 2.26)	43.17 (± 0.53)
NCN	35.14 (± 1.04)	53.41 (± 1.46)	13.22 (± 0.56)	65.82 (± 2.66)	20.84 (± 1.31)	81.89 (± 0.31)	53.76 (± 0.20)
NCNC	36.70 (± 1.57)	53.72 (± 0.97)	18.81 (± 1.16)	>24h	20.49 (± 3.97)	82.24 (± 0.40)	51.69 (± 1.48)
NBFNet	31.12 (± 0.75)	31.39 (± 1.34)	>24h	>24h	OOM	OOM	OOM
PEG	36.03 (± 0.75)	45.56 (± 1.38)	8.70 (± 1.26)	50.12 (± 6.55)	18.29 (± 1.06)	OOM	OOM
LPFORMER	33.27 (± 1.33)	51.58 (± 1.83)	22.71 (± 1.30)	35.23 (± 0.37)	23.05 (± 0.16)	84.01 (± 0.10)	57.30 (± 0.50)
SP4LP	38.52 (± 1.19)	66.28 (± 0.63)	23.01 (± 0.39)	47.96 (± 3.82)	20.00 (± 1.20)	76.9 (± 1.11)	55.45 (± 0.92)

Table 5: Hits@K (%) results across all datasets, following the HeaRT evaluation setting [Li et al. \(2023\)](#). The top three results for each metric are highlighted using **first**, **second**, and **third**. OOM indicates that the model ran out of memory, while >24h denotes that the method did not complete within 24 hours.

Models	Cora		Citeseer		Pubmed	
	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10
<i>GNN + SP len.</i>	14.21 (± 1.44)	33.43 (± 2.69)	20.90 (± 0.79)	47.82 (± 1.11)	7.12 (± 0.41)	5.63 (± 0.52)
<i>Sequence Model</i>	16.86 (± 1.26)	36.03 (± 1.75)	27.45 (± 1.55)	54.20 (± 2.35)	8.58 (± 0.75)	12.87 (± 0.85)
SP4LP	17.27 (± 0.57)	38.52 (± 1.19)	41.08 (± 1.84)	66.28 (± 0.63)	10.87 (± 0.31)	23.01 (± 0.39)

Table 6: Ablation study results (%). MRR and Hits@K with mean and std. deviations over 5 runs with different seeds.

den dimensions $\in \{32, 64, 128, 256\}$ and prediction layers $\in \{1, 2, 3\}$. For large-scale datasets, we follow the reduced search space adopted in [Li et al. \(2023\)](#) to avoid excessive compute. For SP4LP, we additionally explore the choice of GNN component $\in \{\text{GCN}, \text{GraphSAGE}, \text{GAT}\}$ and sequence model $\in \{\text{LSTM}, \text{Transformer}\}$, the best models are shown in Table 8. The best hyperparameters are selected based on validation performance. All reported metrics are averaged over 5 different seeds.

Shortest path calculation for SP4LP For the computation of shortest paths between node pairs u and v , we used the `shortest_path` function from the `networkx` library ([Hagberg et al. 2008](#)). This function returns a single shortest path between two nodes to ensure computational efficiency, even when multiple shortest paths exist. If no path was found between u and v (i.e., they belonged to different connected components), we assigned a synthetic path of length one directly connecting u and v .

In our implementation of SP4LP, the sequential encoder can be instantiated as a Transformer with learnable positional encodings. We use PyTorch’s `TransformerEncoder`, composed of `TransformerEncoderLayer` blocks with 4-head self-attention, feedforward sublayers, ReLU activation, and dropout. We tune the number of layers $\in \{1, 2\}$, attention heads $\in \{2, 3, 4\}$, and feedforward dimensionality $\in \{32, 64, 128\}$. A trainable positional embedding matrix is added to node embeddings to preserve path order. Variable-length paths are handled via a source key padding mask, and the output is aggregated using masked mean pooling followed by layer normalization. The resulting path representation is then combined with the GNN-derived embeddings of the source and target nodes to compute the final link score.

For the LSTM-based encoder, we implement both unidirectional and bidirectional variants using PyTorch’s `nn.LSTM`. Input sequences are packed with `pack_padded_sequence` to handle variable-length paths. We tune the number of layers $\in \{1, 2\}$ and hidden size $\in \{32, 64, 128\}$. The final hidden

Dataset	Cora	Citeseer	Pubmed	ogbl-collab	ogbl-ddi	ogbl-ppa	ogbl-citation2
#Nodes	2708	3327	18717	235868	4267	576289	2927963
#Edges	5278	4676	44327	1285465	1334889	30326273	30561187

Table 7: Dataset statistics.

Dataset	GNN model	Sequence model
Cora	GCN	Transformer
Citeseer	GCN	LSTM
Pubmed	SAGE	Transformer
ogbl-collab	SAGE	Transformer
ogbl-ddi	GCN	Transformer

Table 8: Best GNN and sequence models selected via hyperparameter tuning.

state (or the concatenation of forward and backward states in the bidirectional case) is used as the path representation and combined with the GNN-based node embeddings for link prediction. Node embeddings are obtained via a GNN encoder selected from GCN, GAT, or GraphSAGE, depending on the experimental setting.

E COMPUTATIONAL COMPLEXITY

In this section, we analyze the computational complexity of our proposed SP4LP model for link prediction, following the formalism and notation introduced in Wang et al. (2024). This framework allows a direct comparison with prior approaches such as GAE, Neo-GNN, BUDDY, PEG, SEAL, NCN, and NCNC.

Let n be the number of nodes, m the number of edges, Δ the maximum degree, d the dimensionality of node features or embeddings, t the number of target links to predict, and k the length of the shortest path between node pairs (typically $k \ll n$). We express total time complexity as $\mathcal{O}(B + C \cdot t)$, where B is the precomputation cost (independent of t), and C is the per-link cost.

Our SP4LP method follows the GNN-then-Structural-Feature (SF) paradigm Wang et al. (2024), applying a Message Passing Neural Network once across the entire graph to compute node embeddings, then leveraging shortest path extraction combined with sequence modeling for each candidate link.

Following the paradigm, the precomputation cost of SP4LP is:

$$B = \mathcal{O}(n\Delta d + nd^2 + T_{\text{sp}})$$

where: $n\Delta d$ accounts for neighborhood aggregation in the GNN, nd^2 represents linear transformations in GNN layers, T_{sp} is the cost of computing shortest paths (e.g., via BFS).

For sparse graphs ($m = \mathcal{O}(n)$), single-source BFS takes $\mathcal{O}(m)$, and all-pairs shortest paths require $\mathcal{O}(nm)$. This cost can often be amortized or approximated in practice using methods like truncated BFS or landmark-based search Sommer (2014).

The per-link cost is:

$$C = \mathcal{O}(kd + T_{\text{seq}})$$

where kd corresponds to the cost of extracting and aggregating embeddings along a path of length k , and T_{seq} denotes the cost of a sequence model (e.g., LSTM, Transformer) applied to the node embeddings along that path.

Importantly, since both the GNN embeddings and shortest paths can be precomputed, B is incurred only once. Given that k is typically small in practice, C remains low even at scale. Unlike subgraph-based methods such as SEAL, which require per-link GNN inference over extracted neighborhoods, SP4LP performs lightweight sequence modeling on compact paths, supporting efficient batched

inference. Additionally, the framework is flexible: T_{sp} can be cached or approximated (Sommer 2014), and T_{seq} depends on the chosen architecture and path length.

We summarize the complexities in the following table:

Method	B	C
GAE	$n\Delta d + nd^2$	d^2
Neo-GNN	$n\Delta d + nd^2 + n\Delta^l$	$\Delta^l + d^2$
BUDDY	$n\Delta d + nh$	$h + d^2$
SEAL	0	$\Delta^{(l+1)}d + \Delta^l d^2$
NCN	$n\Delta d + nd^2$	$\Delta d + d^2$
NCNC	$n\Delta d + nd^2$	$\Delta^2 d + \Delta d^2$
PEG	$n\Delta d + nd^2 + nD^2$	d^2
SP4LP	$n\Delta d + nd^2 + T_{\text{sp}}$	$kd + T_{\text{seq}}$

Table 9: Comparison of precomputation (B) and per-link (C) complexities across methods.

All methods conform to the form $\mathcal{O}(B + C \cdot t)$, with a one-time graph-level computation and a per-target-link cost. In the table, h is the cost of the hash function used in BUDDY, l denotes the radius of local neighborhoods in Neo-GNN and SEAL, and D denotes the number of Laplacian eigenvectors used for spectral positional encoding in PEG. GAE, NCN, and BUDDY are efficient but limited in expressiveness. NCNC enhances NCN via soft neighbor completion at a slightly higher cost. PEG incorporates spectral encoding ($\mathcal{O}(nD^2)$), while SEAL is the most computationally intensive due to subgraph extraction and per-link GNN processing. SP4LP, in contrast, strikes a balance between efficiency and expressiveness by combining GNN-based embeddings with sequence modeling over shortest paths, whose cost can be mitigated via approximation techniques (Sommer 2014), ensuring scalability for large graphs.