A OUTLINE

A.1 Dataset Description

The description of all datasets is listed below:

Cora, **CiteSeer**, and **PubMed** [65] are three citation network datasets, where nodes represent papers and edges represent citation relationships. The node features are word vectors, where each element indicates the presence or absence of each word in the paper.

Coauthor CS and **Coauthor Physics** [48] are co-authorship graphs based on the Microsoft Academic Graph [56], where nodes are authors, edges are co-author relationships, node features represent paper keywords, and labels indicate the research field.

Amazon Photo and **Amazon Computers** [48] are segments of the Amazon co-purchase graph, where nodes represent items and edges represent that two goods are frequently bought together. Given product reviews as bag-of-words node features.

ogbn-arxiv and **ogbn-papers100M** [30] are two citation graphs indexed by MAG [56]. Each paper involves averaging the embeddings of words in its title and abstract. The embeddings of individual words are computed by running the skip-gram model.

ogbn-products [30] is a co-purchasing network, where nodes represent products and edges represent that two products are frequently bought together. Node features are generated by extracting bag-of-words features from the product descriptions.

Flickr [69] dataset originates from the SNAP, where nodes represent images, and connected images from common properties. Node features are 500-dimensional bag-of-words representations.

Reddit [28] dataset collected from Reddit, where 50 large communities have been sampled to build a post-to-post graph, connecting posts if the same user comments on both. For features, off-the-shelf 300-dimensional GloVe vectors are used.

A.2 Compared Baselines

The backbone GNNs used in experiments are listed below:

GCN [38] introduces a novel approach to graphs that is based on a first-order approximation of spectral convolutions on graphs. This approach learns hidden layer representations that encode both local graph structure and features of nodes.

GAT [53] utilizes attention mechanisms to quantify the importance of neighbors for message aggregation. This strategy enables implicitly specifying different weights to different nodes in a neighborhood, without depending on the graph structure upfront.

GCNII [15] incorporates initial residual and identity mapping. Theoretical and empirical evidence is presented to demonstrate how these techniques alleviate the over-smoothing problem.

GATv2 [3] introduces a variant with dynamic graph attention mechanisms to improve GAT. This strategy provides better node representation capabilities and enhanced robustness when dealing with graph structure as well as node attribute noise.

GraphSAGE [28] leverages neighbor node attribute information to efficiently generate representations. This method introduces a general inductive framework that leverages node feature information to generate node embeddings for previously unseen data.

Cluster-GCN [17] is designed for training with stochastic gradient descent (SGD) by leveraging the graph clustering structure. At each step, it samples a block of nodes that associate with a dense subgraph identified by a graph clustering algorithm, and restricts the neighborhood search within this subgraph.

GraphSAINT [69] is a inductive framework that enhances training efficiency through graph sampling. Each iteration, a complete GCN is built from the properly sampled subgraph, which decouples the sampling from the forward and backward propagation.

ShaDow [68] decouples the depth and scope of GNNs for informative representations in node classfication. This approach propose a design principle to decouple the depth and scope of GNNs – to generate representation of a target entity, where a properly extracted subgraph consists of a small number of critical neighbors, while excluding irrelevant ones.

SGC [59] simplifies GCN by removing non-linearities and collapsing weight matrices between consecutive layers. Theoretical analysis show that the simplified model corresponds to a fixed low-pass filter followed by a linear classifier.

APPNP [39] leverages the connection between GCN and PageRank to develop an enhanced propagation method. This strategy leverages a large, adjustable neighborhood for classification and can be easily combined with any neural network.

PPRGo [2] proposes an efficient approximation of diffusion in GNNs for substantial speed improvements and better performance. This approach utilizes an efficient approximation of information diffusion in GNNs resulting in significant speed gains while maintaining competitive performance.

SIGN [26] introduces a novel, efficient, and scalable graph deep learning architecture that eliminates the need for graph sampling. This method sidesteps the need for graph sampling by using graph convolutional filters of different size that are amenable to efficient pre-computation, allowing extremely fast training and inference.

S^2GC [78] introduces a modified Markov Diffusion Kernel for GCN, which strikes a balance between low- and high-pass filters to capture the global and local contexts of each node.

GBP [14] introduces a scalable GNN that employs a localized bidirectional propagation process involving both feature vectors and the nodes involved in training and testing. Theoretical analysis shows that GBP is the first method that achieves sub-linear time complexity for both the pre-computation and the training phases.

AGP [55] proposes a unified randomized algorithm capable of computing various proximity queries and facilitating propagation. This method provides a theoretical bounded error guarantee and runs in almost optimal time complexity.

GAMLP [73] is designed to capture the inherent correlations between different scales of graph knowledge to break the limitations of the enormous size and high sparsity level of graphs hinder their applications under industrial scenarios.

GRAND+ [24] develops the generalized forward push algorithm called GFPush, which is utilized for graph augmentation in a mini-batch fashion. Both the low time and space complexities of GFPush enable GRAND+ to efficiently scale to large graphs.

Table 5: The statistical information of the experimental datasets.

Dataset	#Nodes	#Features	#Edges	#Classes	#Train/Val/Test	#Task	Description
Cora	2,708	1,433	5,429	7	140/500/1000	Transductive	citation network
CiteSeer	3,327	3,703	4,732	6	120/500/1000	Transductive	citation network
PubMed	19,717	500	44,338	3	60/500/1000	Transductive	citation network
Amazon Photo	7,487	745	119,043	8	160/240/7,087	Transductive	co-purchase graph
Amazon Computer	13,381	767	245,778	10	200/300/12,881	Transductive	co-purchase graph
Coauthor CS	18,333	6,805	81,894	15	300/450/17,583	Transductive	co-authorship graph
Coauthor Physics	34,493	8,415	247,962	5	100/150/34,243	Transductive	co-authorship graph
ogbn-arxiv	169,343	128	2,315,598	40	91k/30k/48k	Transductive	citation network
ogbn-products	2,449,029	100	61,859,140	47	196k/49k/2204k	Transductive	co-purchase graph
ogbn-papers100M	111,059,956	128	1,615,685,872	172	1200k/200k/146k	Transductive	citation network
Flickr	89,250	500	899,756	7	44k/22k/22k	Inductive	image network
Reddit	232,965	602	11,606,919	41	155k/23k/54k	Inductive	social network

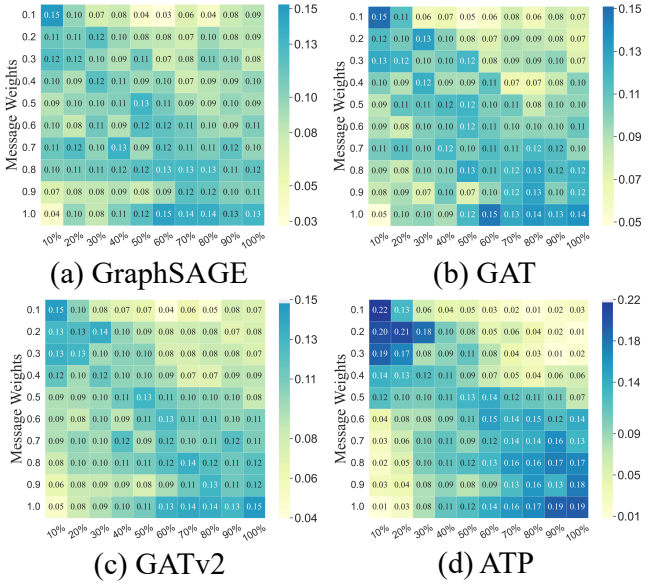

Figure 6: Comparison of the attention- and LNC encoding-based message aggregation weights (similar to propagation kernel coefficients) on ogbn-arxiv. The x-axis represents the ranking of node degrees from low to high order.

A.3 Experiment Environment

Experiments are conducted with Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz, and a single NVIDIA GeForce RTX 3090 with 24GB GPU memory. The operating system of the machine is Ubuntu 20.04.5 with 768GB of memory. As for software versions, we use Python 3.8.10, Pytorch 1.13.0, and CUDA 11.7.0.

A.4 Weighted Aggregation and LNC Encoding

In Sec. 4.2, we provide a brief discussion of potential scalability concerns associated with the attention mechanisms in web-scale graph learning scenarios. We also highlight the incompatibility of these end-to-end learnable message aggregation mechanisms with the LNC encoding introduced in ATP for optimizing propagation kernel coefficients r. To delve deeper into these statements and present a comprehensive evaluation of weighted message aggregation and LNC encoding within ATP, this section begins by clarifying the distinctions and connections between learnable attention mechanisms and graph propagation equations. Then, we present visual experimental results for both ATP and end-to-end attention-based approaches, including GraphSAGE, GAT, and GATv2.

Graph Attention Mechanisms. To improve the predictive performance of the current node i, GraphSAGE proposes to explicitly consider its first-order neighbors, denoted as $j \in \mathcal{N}_i$. Specifically, during the message aggregation, GraphSAGE treats all neighbors with equal importance (indiscriminated aggregation), with a key aspect being the combination of aggregated messages using an end-to-end learnable mechanism. This can be formalized as follows

$$\mathbf{X}_u = \text{Aggregate}\left(\mathbf{W}, \mathbf{X}_u, \{\mathbf{X}_v, \forall v \in \mathcal{N}(u)\}\right). \tag{12}$$

To achieve fine-grained message aggregation for each node, GAT employs a d-dimension embedding-based learnable scoring function with trainable $\mathbf{a}$, $\mathbf{W}$, denoted as $e : \mathbb{R}_q^d \times \mathbb{R}_v^d \to \mathbb{R}$, to obtain the attention score α of each "key" neighbor in generating representations for the current "query" node (i.e., attention mechanism).

$$\begin{aligned} e\left(\mathbf{X}_i, \mathbf{X}_j\right) &= \text{LeakyReLU}\left(\mathbf{a}^\top \cdot \left[\mathbf{W}\mathbf{X}_i \| \mathbf{W}\mathbf{X}_j\right]\right), \\ \alpha_{ij} &= \text{softmax}\left(e\left(\mathbf{X}_i, \mathbf{X}_j\right)\right) = \frac{\exp\left(e\left(\mathbf{X}_i, \mathbf{X}_j\right)\right)}{\sum_{j \in \mathcal{N}_i} \exp\left(e\left(\mathbf{X}_i, \mathbf{X}_{j'}\right)\right)}. \end{aligned} \tag{13}$$

Then, GAT takes into account neighbor messages with varying scores when generating representations for the current node.

$$\mathbf{X}_u = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot \mathbf{W}\mathbf{X}_j\right). \tag{14}$$

Due to the globally shared learnable parameters in the e, different "query" node embeddings $\mathbf{X}_u \in \mathbb{R}_q^d$ will yield the same score ranking list in extreme scenarios (e.g., complete bipartite graphs). In other words, GAT may disproportionately focus on a fixed "key" neighbor (i.e., static attention), which contradicts the original intention of flexible attention composition. Building upon this observation, GATv2 modifies the order of importance scores computation to achieve a more expressive graph attention mechanism.

$$e\left(\mathbf{X}_i, \mathbf{X}_j\right) = \mathbf{a}^\top \text{LeakyReLU}\left(\mathbf{W} \cdot \left[\mathbf{X}_i \| \mathbf{X}_j\right]\right). \tag{15}$$

Table 6: Model performance with NP optimization strategies.

Models	Computer	Physics	ogbn-arxiv	Flickr
GCNII	82.64±0.5	92.78±1.2	72.68±0.3	53.11±0.1
GCNII+ATP	83.75±0.4	**94.32±1.0**	**75.37±0.2**	**53.96±0.1**
GNN-LSPE	83.34±0.5	93.90±1.4	72.96±0.3	52.24±0.1
Graphormer	82.95±0.6	93.54±1.3	72.35±0.3	51.86±0.2
Gophormer	83.10±0.5	93.67±1.1	72.60±0.2	52.28±0.1
NAGphormer	83.76±0.5	93.85±1.2	73.75±0.3	53.40±0.2
LiteGT	82.84±0.6	93.12±1.5	73.13±0.3	52.33±0.1
SAT	83.55±0.5	94.12±1.2	73.84±0.3	52.57±0.1
AGT	**83.84±0.6**	93.88±1.1	73.98±0.3	53.24±0.2

Reviewing graph attention, we find that their optimization can also be derived from a node-wise perspective. For instance, GAT aims to identify neighbors during aggregation, while GATv2 adopts fine-grained attention score modeling. Fundamentally, graph attention represents a specific instance within the broader context of graph propagation equations, customizing message aggregation strategies for each node in an end-to-end learnable manner (i.e., node-pair based w in Eq. (3)). Intuitively, graph attention is effective, but the intricate topology in web-scale graphs brings unique challenges. According to Table 1, it is evident that current attention mechanisms struggle to maintain effectiveness and consistency, let alone provide the scalability required for large-scale graph learning. To illustrate this issue, we provide the following visual analysis.

Visual Analysis. We report the visual results in Fig. 6, where the x-axis indicates the node set with degrees within the Top 10% of the ranking from low to high order and the heat map value represents the percentage of nodes in the set that have achieved the correspondent message weights. For ATP, the y-axis is the node-adaptive r. For others, we first train each model and then obtain the average attention score α for each node in first-order aggregation.

Building upon this, we draw the following conclusions: (1) The performance of ATP aligns with the key intuition derived from our empirical study in Sec.1. Specifically, nodes with smaller degrees tend to have smaller r, leading to a more inclusive knowledge acquisition from their neighbors, while nodes with larger degrees have larger r to enable fine-grained discrimination of neighbors. (2) Progressing from GraphSAGE to GATv2, we observe that their optimization objectives align with the pre-process in ATP. However, as previously emphasized, when faced with intricate topologies in web-scale graphs, existing methods cannot fully capture potential structural patterns through learnable mechanisms. The lack of distinct color differentiations leads to their sub-optimal performance.

A.5 Graph Transformers and GCNII-based ATP

We commence by revisiting graph transformer and graph attention from the perspective of the *attention* mechanism. Then, we elucidate the distinctions between ATP and graph transformer. With experimental analysis, we discuss if ATP is preferred to perform graph propagation on web-scale graphs comparing graph transformer.

Graph Transformer Mechanisms. From a self-attention perspective, the graph attention mechanism calculates only the first-order neighbors of the current node, while the graph transformer considers all nodes within the graph. From a message aggregation perspective, graph transformer correspond to fully connected dense graphs, while graph attention corresponds to a relatively sparse graph. In terms of structural encoding, graph transformers provide the model with high-dimensional global structural positional priors, whereas graph attention focuses more on the local neighbors.

Specifically, transformers consist of multiple transformer layers, with each comprising a self-attention module and a feed-forward network (FFN). Considering the l-th transformer layer, the input features $\mathbf{H}^{(l-1)} \in \mathbb{R}^{N\times d}$ (where $\mathbf{H}^{(0)} = \mathbf{X}^{(0)}$) are initially transformed using three weight matrices $\mathbf{W}^Q \in \mathbb{R}^{d\times d_Q}, \mathbf{W}^K \in \mathbb{R}^{d\times d_K}, \mathbf{W}^V \in \mathbb{R}^{d\times d_V}$ to generate the corresponding query, key, and value matrices $Q, K, V \in \mathbb{R}^{N\times d}$. For simplicity, we assume that $d = d_K = d_Q = d_V$. The formulation of the transformer layer is then as follows:

$$\mathbf{Q} = \mathbf{H}^{(l-1)}\mathbf{W}^Q, \mathbf{K} = \mathbf{H}^{(l-1)}\mathbf{W}^K, \mathbf{V} = \mathbf{H}^{(l-1)}\mathbf{W}^V$$
$$\mathbf{B}^{(l)} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}, \quad \mathbf{H}^{(l)} = \mathrm{FFN}\left(\mathrm{softmax}\left(\mathbf{B}^{(l)}\right)\mathbf{V}\right). \tag{16}$$

Graph transformers have significantly advanced the field of graph learning by addressing fundamental limitations inherent in GNNs, such as over-smoothing. Building upon this, graph transformers empower nodes to incorporate information from any other nodes in the graph, thereby overcoming the constraints of the limited receptive field. In other words, the fundamental concept behind graph transformers is to incorporate structural information into the transformer architecture in a learnable manner, facilitating node predictions.

Related Works. To compare the performance improvement brought by ATP to GCNII with models based on the graph transformer architecture, we summarize the key characteristics of representative graph transformers proposed in recent years as follows

LSPE [23] proposes a generic architecture to decouple node attributes and topology in a learnable manner for better performance. This method proposes to decouple structural and positional representations to learn these two essential properties.

Graphormer [67] utilizes node degree and neighborhood-based spatial centrality to combine additional topological structure information in the learnable message aggregation process.

Gophormer [77] utilizes well-designed sampled ego-graphs, introduces a proximity-enhanced transformer mechanism to capture structural biases for better aggregation. Meanwhile, this strategy considers the stability in training and testing.

LiteGT [7] introduces an efficient graph transformer architecture that incorporates sampling strategies and a multi-channel transformer mechanism with kernels for better performance.

SAT [10] employs various graph learning models to extract correlated structural information within the current node's neighborhood, including utilize graph Laplacian eigenvectors-based encoding mechanism to improve transformer architectures.

AGT [43] consists of a learnable centrality encoding strategy and a kernelized local structure encoding mechanism to extract structural patterns from the centrality and subgraph views to improve node representations for the node-level downstream tasks.

NAGphormer [11] treats each node as a sequence containing a series of tokens. For each node, NAGphormer aggregates the neighborhood features from different hops into different representations.

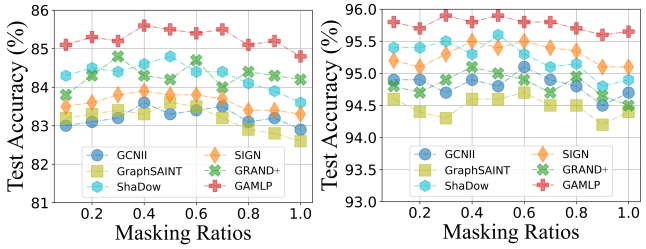

Figure 7: Performance under the influence of masking ratio.

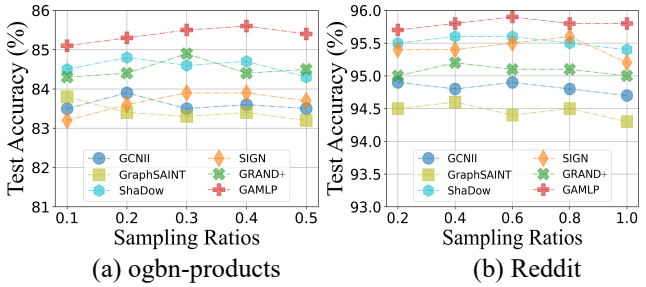

Figure 8: Performance under the influence of sampling ratio.

Experimental Analysis. we present the predictive performance of graph transformers and GCNII-based ATP, in Table 6. Notably, we opt for GCNII over GCN to ensure a fair comparison, as the simple computations in GCN appear obsolete compared with well-designed transformer mechanisms. Based on the results, we observe that graph transformers exhibit a significant advantage on small-scale datasets such as Computer and Physics. This is attributed to their ability to effectively capture the simple and direct topological structures. Conversely, graph transformers struggle to perform well as the dataset size grows due to the increasingly intricate topology, as evidenced by their performance on ogbn-arxiv and Flickr.

Graph Attention/Transformer vs ATP. Fundamentally, both graph attention and graph transformer share the core idea of achieving better message aggregation through end-to-end learnable mechanisms. However, graph attention pays more attention to local neighbors, whereas graph transformers aim to encode global topology. Although GATv2 and NAGphormer enhance the expressiveness of score function in graph attention and improve local representations in graph transformers, they both exhibit significant disadvantages as follows : (1) The representation capacity of end-to-end learnable mechanisms is contentious, especially in facing the intricate topology of web-scale graphs. In other words, the debate about whether they can successfully capture the LNC of each node remains uncertain. Our experiments on web-scale graphs have yielded unsatisfying results. (2) Complex model architectures with vast learnable parameters lead to scalability issue. Although NAGphormer can be trained on ogbn-papers100M with the use of sampling and mini-batch training strategies, it remains challenging to deploy and is prone to instability. It is highly sensitive to sampling results and training hyperparameters such as token size.

To address these issues, we introduce LNC, which offers a comprehensive node characterization based on *features*, *positions* in the graph, and *local topological structure*. As shown in our empirical study in Sec. 1, LNC reveals key insights for achieving robust node classification performance on web-scale graphs with intricate topology. Building upon this, we propose ATP to improve graph propagation equations Eq. (3), which seamlessly combines node *features*, *positions* (from a global perspective, centrality-based position encoding similar to graph transformer), and *local topological structure* (from a local perspective, connectivity-based local topological structure encoding similar to graph attention). Meanwhile, as a weight-free and plug-and-play strategy, ATP improves the running efficiency of the most of existing GNNs.

A.6 Masks in High-bias Propagation Correction

HPC first samples a subset of nodes $\tilde{\mathcal{V}} \subset \mathcal{V}$. The design principles for the sampling mechanism are as follows: (1) selecting all nodes that rank in the Top-θ% based on their degree when nodes are sorted from highest to lowest degree. This is to strike a balance between the optimal convergence radius and over-smoothing from a global graph propagation perspective. (2) Selecting partial nodes outside the above node degree rankings. Specifically, for other relatively sparse nodes, we perform random sampling with a fixed sampling ratio of 0.2. Similar to dropout in training process, this strategy aims to enhance the robustness of node representations from a regularization perspective while further reducing the pre-computation and training costs associated with topology. Subsequently, HPC applies edge masking to the node set $\tilde{\mathcal{V}}$ by the mask token $[M]$. It is worth noting that we have provided experimental analysis into how different values of θ impact the performance improvement brought by High-Deg selection in Fig.2. Therefore, in this section, we supplement the discussion of the effect of the edge masking ratio $[M]$ performed by HPC on predictive performance.

According to the experimental results presented in Fig. 7, in the transductive setting, increasing the masking ratio from zero has an overall positive impact on predictive performance. However, when the masking ratio becomes excessively high, the performance deteriorates when handling edge sparsity. Furthermore, in the inductive setting, we find that lower masking ratios may have a negative effect, in stark contrast to the results in the transductive setting. The observed variation arises from the inductive setting's demand for richer neighborhood knowledge in predicting unseen nodes. Nevertheless, as we increase the masking ratio, the benefits of eliminating potential high-bias propagation outweigh the drawbacks of reduced neighborhood knowledge. In conclusion, based on the empirical analysis of the experimental results mentioned above, we recommend setting the masking ratio to 0.5, as it tends to yield optimal predictive performance in most cases.

Similar to the conclusions drawn from Fig.2 and Fig.7, the experimental results in Fig. 8 indicate that randomly sampling too many relatively sparsely connected nodes for HPC can adversely affect node prediction performance due to significant topological information gap. Conversely, selecting an appropriate sampling ratio can strike a balance between mitigating potential high-bias propagation and topological gap, thereby contributing to significant performance improvements in the original backbone.

Table 7: FSGNN performance gain with ATP.

Models	arxiv	products	100M	Flickr	Reddit
FSGNN	73.5±0.3	80.8±0.2	66.2±0.2	53.4±0.2	94.1±0.1
FSGNN+ATP	75.3±0.3	84.5±0.2	69.9±0.2	55.1±0.1	95.9±0.1

Table 8: HPC performance with different strategies.

Models	arxiv	products	papers100M
SGC	71.84±0.26	75.92±0.07	64.38±0.15
SGC+Sampling	71.59±0.34	76.14±0.15	64.24±0.27
SGC+ATP	72.25±0.20	76.80±0.08	64.56±0.19
APPNP	72.34±0.24	78.84±0.09	65.26±0.18
APPNP+Sampling	72.25±0.40	78.60±0.24	65.14±0.26
APPNP+ATP	72.76±0.17	79.46±0.11	65.48±0.20
GRAND+	73.86±0.28	79.55±0.20	66.86±0.17
GRAND++Sampling	73.58±0.46	79.47±0.36	66.70±0.28
GRAND++ATP	74.10±0.22	80.07±0.14	67.12±0.16

Table 9: Efficiency results for sampling-based methods.

Models	products	Reddit
GraphSAGE	2736s	70.2s
Cluster-GCN	2183s	62.7s
GraphSAINT	1892s	51.5s
ShaDow	1624s	46.8s
GAMLP+ATP	1974s	49.3s
GAMLP+ATP/Eigen	1480s	41.6s

Table 10: Performance with different LNC strategies.

Model	arxiv	products
GAMLP	73.43±0.32	81.41±0.22
+ ATP/E	75.69±0.30	84.85±0.28
+ ATP/E w PageRank	75.64±0.39	84.96±0.25
+ ATP/E w Katz	75.78±0.36	84.90±0.34
+ ATP/E w both	75.84±0.29	85.12±0.26

A.7 Experiments for Comprehensive Evaluation

It is worth noting that, in addition to the baseline backbone models compared in Sec. 4.2, FSGNN [45] stands out as a recently proposed scalable GNN model based on a decoupling strategy. Leveraging Soft-Selector and Hop-Normalization, FSGNN constructs robust neural predictors from the perspective of node features, exhibiting significant advantages in node-level prediction tasks. To further explore whether ATP can enhance the performance of FSGNN, we present its experimental results on transductive and inductive datasets in Table 7. Based on these findings, we consistently observe that ATP enhances the performance of FSGNN across these 5 different datasets. This improvement stems from the fine-grained propagation results achieved by ATP, which effectively decouple feature generation and representation learning within FSGNN.

In our proposed ATP, the HPC module adopts a targeted regularization approach focusing on High-Deg. Although GraphSAGE employs random neighborhood sampling akin to our approach, the key disparity lies in ATP's regularization objective centered on High-Deg, while GraphSAGE uniformly conducts dropout. To elucidate these distinctions and provide a comprehensive evaluation, we present experimental results in Table 8. In contrast to GraphSAGE's uniform neighborhood sampling strategy at each layer, which disregards node degree discrepancies and applies identical sampling proportions to all nodes, ATP incorporates tailored propagation rules based on node connectivity. As demonstrated in our theoretical analysis in Sec. 3.1, customized propagation rules should prioritize High-Deg. However, GraphSAGE's uniform approach leads to inconsistent performance across experiments, falling short of our proposed method. Furthermore, GraphSAGE exhibits higher variance due to its inability to consistently produce high-quality sampling outcomes across random runs. Conversely, ATP leverages insights from our theoretical analysis to adapt propagation rules for High-Deg, resulting in enhanced consistency and predictions.

As discussed in Sec. 3.2, the eigenvector-based position encoding is designed to exploit the spectral domain perspective, offering a more precise description of the node's LNC. However, it entails spectral decomposition, which may incur computational overheads. While approximate estimation methods can partially alleviate these costs, they still represent a non-negligible computational overhead. Therefore, we proceed to introduce feasible alternative solutions in the subsequent sections. It is noteworthy that we present an intuitive efficiency analysis in Table 9 to further substantiate our viewpoint. Alternatively, for minimizing computational overheads while optimizing NP, we suggest exploring alternative designs that enhance spatial domain encoding of local context. Utilizing PageRank or Katz centrality based on degree-based position encoding presents viable alternatives to eigenvector-based position encoding. These methods offer lower computational costs compared to eigenvector centrality. However, they share fundamental similarities with degree-based position encoding, thereby limiting the improvements they can provide over using ATP alone. Detailed experimental results are shown in Table 10.