# MAKE EACH STEP COUNT: ENTROPIC WALKER FOR COMPACT AND INFORMATIVE GRAPH REPRESENTATION LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Learning compact and predictive representations that capture critical substructures is fundamental for effective graph learning. However, existing works either risk critical information loss or suffer from the inclusion of task-irrelevant distractors. In this work, we present an information-theoretic formulation for compact graph learning and introduce a tractable, explicit proxy for quantifying structural redundancy. Concretely, our formulation decouples structural complexity from the mutual-information term that is entangled with subgraph diversity, and further aligns the graph compactness objective with the entropy of random walk sequences. This not only avoids direct mutual-information estimation, but also yields an explicit and tractable objective for learning compact predictive substructures. Building on this formulation, we propose Entropic Walker (EW), a framework for learning compact yet predictive graph representations via entropy-controlled random walks, with scalable formulations for both graph-level and node-level tasks. Experiments on 15 benchmarks across diverse applications show consistent performance gains and strong ablation results, highlight the need to rethink the way to measure structural redundancy of graph data.

## 1 INTRODUCTION

Graphs have emerged as a powerful abstraction for representing diverse real-world data. The ubiquity of graph-structured data has fueled a surge of research in graph representation learning, particularly in domains such as biochemistry Zhang et al. (2025b), social networks Fan et al. (2019), and neuroscience Behrouz et al. (2024).

Despite these advances, existing studies primarily focus on improving graph neural architectures to fit rapidly increasing graph data Rampásek et al. (2022); Behrouz & Hashemi (2024); Wang et al. (2024a); Zhang et al. (2025a), while overlooking a fundamental challenge that real-world graphs inevitably include redundant or even misleading signals. Consequently, noise or task-irrelevant details can be broadcast along spurious edges, obscuring critical information, relationships, and generating untrustworthy results.

To address this, conventional approaches can be broadly divided into statistical and sparsification-based methods. The former aims to identify representative substructures or densely connected subgraphs through predefined statistical criteria Fang et al. (2019a); Gionis & Tsourakakis (2015); Yan & Han (2002); Ketkar et al. (2005b); Zaki (2005), while the latter performs iterative pruning to simplify graph topology Rong et al. (2020); Zheng et al. (2020); Chen et al. (2018). However, these methods either lack task-awareness or rely on heuristic design, suffering from limited scalability and risk of information loss.

Another line of research follows the conventional information bottleneck (IB) paradigm Tishby et al. (2000); Alemi et al. (2017), seeking to derive compact graph representations by optimizing mutual information objectives Wu et al. (2020); Zhang et al. (2025b); Ding et al. (2024a;b); Yu et al. (2021; 2022); Hoang & Lee (2025). Despite the conceptual appeal, these methods present notable limitations. Specifically, graph data are discrete, non-Euclidean, and highly interdependent, which makes mutual information difficult to define and estimate in a tractable manner. Moreover, prevailing mutual information estimators are mainly established on simplifying assumptions and thus their



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

variational approximations often introduce a non-negligible estimation gap, making the induced objectives difficult to optimize robustly in practice.

In this paper, we propose a new strategy for compact and predictive graph learning inspired by the coding theorem Shannon (1948); Rosvall & Bergstrom (2008): a chaotic system (*e.g.*, an uniform distribution or a graph rich in redundancy) requires more bits to encode each element, thereby increasing the average code length necessary for describing their interrelationships and consequently leading to higher prediction entropy. That is, the structural complexity of a graph can be quantified by the entropy associated with predicting transitions along its topology. Conversely, minimizing such entropy effectively reduces the structural redundancy within the graph.

Building on this foundation, structural complexity can be conceptualized as uncertainty in predicting transitions within a graph. This motivates the use of transition entropy as an explicit and tractable proxy for reducing graph redundancy, thereby enabling a principled formulation for compact yet predictive graph learning. Moreover, by decoupling structural complexity from subgraph diversity, we present a transition-based characterization of graph compactness that avoids direct mutual-information estimation and aligns the minimality objective more naturally with the goal of graph learning.

Furthermore, we propose Entropic Walker (EW), which integrates the above theoretical formulation into a unified framework through a cohesive design, yielding a flexible, scalable, and tractable solution for learning succinct yet predictive representations.

We empirically validate the proposed strategy through extensive experiments on 15 benchmark datasets spanning diverse graph learning tasks. The results show that our approach achieves strong overall performance and delivers notable gains over state-of-the-art models, particularly on large-scale and structurally complex graphs.

The main contributions of this work are summarized as follows:(1) We present an information-theoretic formulation for compact and predictive graph learning. By characterizing structural complexity through transition entropy, our formulation avoids direct mutual-information estimation and aligns the minimality objective more naturally with graph learning. (2) We develop Entropic Walker (EW), a unified framework that instantiates the proposed formulation in a tractable and scalable manner for both graph-level and node-level tasks. (3) We extensively evaluate our method on diverse applications, where the empirical results demonstrate strong performance and validate the effectiveness of our modeling strategy.

## 2 METHOD

### 2.1 BACKGROUND

**Compact Graph Learning.** Existing compact graph learning methods seek to preserve task-relevant structures while removing superfluous components through learning-based sparsification, condensation or pooling, while information-bottleneck approaches further formalize this goal through the principle of minimal sufficiency. EW shares this principle but characterizes graph compactness from a transition-based entropy perspective rather than through direct mutual-information estimation. A detailed discussion of related work is provided in Appendix 8.

**Random Walk for Graph Learning.** Existing random-walkers primarily exploit walks to capture structural patterns or model long-range dependencies. We revisit random walks from a simple yet distinct perspective by formulating transition uncertainty as a tractable surrogate for graph structural compactness, thereby promoting parsimonious yet predictive graph representations.

### 2.2 NOTATIONS

Throughout this section, we denote a graph by $G = \{V, E, X, Z\}$, where $V$ is the set of nodes, $E$ is the set of edges, $X \in R^{|V|\times d}$ and $Z \in R^{|E|\times d'}$ denote the feature matrix of the nodes and edges, respectively. A walk $S_k$ of length $L$ on graph $G_k$ is defined as a sequence of nodes connected by edges, *i.e.*, $S_k = \{v_1^k, ..., v_L^k\} \in V^L$ such that $\forall t \in [L-1], (v_t, v_{t+1}) \in E$. Each walk induces a subgraph $G_k'$, obtained via a deterministic mapping $\mathcal{F} : S \to G'$, where $V(G') = \{v_t | v_t \in S_k\}$, $E(G') = \{(v_t, v_{t+1})\}_1^{L-1}$.

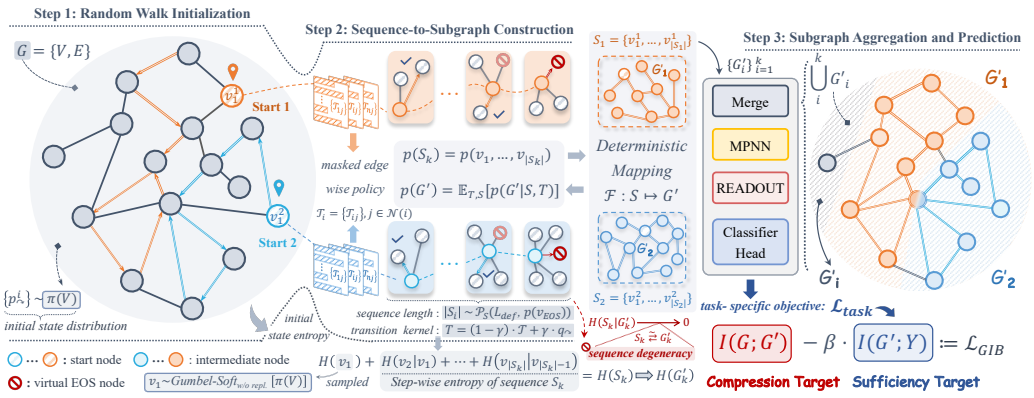



Figure 1: Overview of the proposed Entropic Walker framework. Step 1: random walk initialization via a learnable scoring function. Step 2: transitions are sampled by differentiable policy heads and converted to subgraphs. Step 3: subgraphs are merged into a unified graph, which is then processed by message-passing, readout, and prediction layers.

### 2.3 Reformulation from Information-Theoretic Perspective

A key motivation of Entropic Walker stems from the coding theory Shannon (1948); Rosvall & Bergstrom (2008), where the structural complexity can be quantified by the entropy of predicting interrelationships among nodes. This naturally leads to a random-walk-based modeling strategy as the uncertainty in selecting the next step serves as a proxy for reducing redundancy within a graph[1].

Formally, let $S$ be a walk sequence of graph $G$, obtained from a sampler $\theta$. The subgraph $G$ is then deterministically constructed by $\mathcal{F}$, which includes all nodes in $S$ together with their associated edges. On this basis, we first consider the mutual information between $G$ and $G'$:

$$I(G;G') = H(G') - H(G'|G), \tag{1}$$

where $H(G')$ serves as an explicit surrogate of information volume, while $H(G'|G)$ denotes the subgraph diversity induced by the intrinsic structure of $G$ and the downstream task. In particular, Eq. (1) suggests minimizing $I(G;G')$ is not identical to learning a compact yet predictive subgraph, as it mixes the complexity of the sampled subgraphs and existence of multiple plausible predictive subgraphs supported by $G$.

Therefore, instead of directly optimizing $I(G;G')$, we focus on $H(G')$ as a more appropriate minimality target, and further derive the following decomposition via the chain rule:

$$I(G';S) = H(G') - H(G'|S) = H(S) - H(S|G'). \tag{2}$$

Since $S \to G'$ is deterministic, we have $H(G'|S) = 0$. Meanwhile, $H(S|G')$ quantifies the ambiguity of recovering walk sequences from an induced subgraph. While theoretically nonzero, our empirical analysis suggests that this term is practically negligible under the considered setting, as detailed in the appendixes. Building on this foundation, we derive:

$$H(G') \approx H(S), \tag{3}$$

provides a tractable surrogate characterization of the subgraph minimality target through the entropy of walk sequences. Accordingly, we optimize

$$\min_{\theta} H(S), \tag{4}$$

which encourages compactness while preserving meaningful subgraph diversity. From this perspective, EW shares with GIB the principle of minimal sufficiency, while departing from the mutual-information formulation by characterizing graph compactness through structural entropy. Next, we derive $H(S)$ and present two complementary formulations for graph-level and node-level tasks.

[1]Based on Shannon (1948), a more complex graph requires more bits to encode its structure, thereby increasing the uncertainty in selecting the next step.

### 2.4 FORMULATION VIA RANDOM WALK

Random walks provide a natural Markovian mechanism to generate sequences from a graph, which factorizes the probability of $S$ as:

$$p(S) = p(v_1, ..., v_L) = p(v_1) \prod_{t=2}^{L} p(v_t|v_{t-1}). \tag{5}$$

On this basis, $H(S)$ is formulated as:

$$H(S) = -\sum_{S} p(S) \log p(S) = -\sum_{S} p(S) \left[ \log p\left(v_1\right) + \sum_{t=2}^{L} \log p\left(v_t|v_{t-1}\right) \right]. \tag{6}$$

By marginalizing over $\{v_2, \ldots, v_L\}$, the first term reduces to entropy of initial distribution $H(v_1)$, given by:

$$-\sum_{S} p(S) \log p(v_1) = -\sum_{v_1} \log p(v_1) \sum_{v_2,\ldots,v_l} p(S) = -\sum_{v_1} p(v_1) \log p(v_1) = H(v_1). \tag{7}$$

The second term in Eq. (6) is obtained by fixing time step $t$ and marginalizing the rest of $S$:

$$\begin{aligned} & -\sum_{v_1,\ldots,v_L} p(v_1, ..., v_L) \log p\left(v_t|v_{t-1}\right) \\ = & -\sum_{v_t, v_{t-1}} \log p\left(v_t|v_{t-1}\right) \sum_{\substack{v_1,\ldots,v_{t-2}, \\ v_{t+1},\ldots,v_L}} p(v_1, ..., v_L) \\ = & -\sum_{v_t, v_{t-1}} p(v_{t-1})\, p\left(v_t|v_{t-1}\right) \log p\left(v_t|v_{t-1}\right) = H(v_t|v_{t-1}). \end{aligned} \tag{8}$$

Substituting Eq. (7) and Eq. (8) into Eq. (6), we derive:

$$H(S) = H(v_1) + \sum_{t=2}^{L} H(v_t|v_{t-1}). \tag{9}$$

Here, $H(v_1)$ and $H(v_t|v_{t-1})$ measure the uncertainty of the initial choice and subsequent transitions, respectively. Together, they serve as the proxy for the structural complexity of $G$, as superfluous signals increase the uncertainty in producing $S$. Hence, optimizing Eq. (9) is essentially identifying a minimal set of traversals to succinctly represent $G$, in line with the preceding discussion.

Though Eq. (9) provides a flexible and efficient formulation for graph-level tasks, it is less suited to node-level tasks for two main reasons: (i) node-level tasks requires global constraint on message passing to reduce redundancy propagation, rather than identifying a specific optimal region or sequence; (ii) node-level prediction necessarily involves all nodes, which conflicts with the objective of subgraph detection or graph compression.

To address this, we consider the average entropy per step $\bar{H}_s$, given by:

$$\bar{H}_s = -\sum_{v_i \in V} p(v_i) \sum_{v_j \in V} p(v_j|v_i) \log p(v_j|v_i). \tag{10}$$

Eq. (10) defines a global objective that quantifies structural complexity across all transitions within the graph. Optimizing it effectively obscures those spurious edges and thus prevents task-irrelevant information from being broadcast, while preserving all nodes for downstream prediction, thereby aligning naturally with node-level tasks.

On this basis, the compression target for node-level tasks can be given as:

$$H(S) = H(v_1) + (L-1) \cdot \bar{H}_s. \tag{11}$$

Here, $(L-1)$ rescales Eq. (11) to the entropy of an $L$-length sequence, ensuring dimensional consistency with Eq. (9).

Collectively, Eq. (9) and Eq. (11) define two complementary forms of graph information bottleneck, which identify critical regions through specific walk sequences and eliminate redundant edges from a global perspective, respectively. Next, we present Entropic Walker, which integrates the above formulations into a unified framework and establishes $H(S)$ as an effective surrogate for $I(G;G)$.

### 2.5 ENTROPIC WALKER

As illustrated in Fig. 1, Entropic Walker consists of three sequential steps.

**Step 1: Random Walk Initialization.** This step specifies the initial state distribution. Specifically, a trainable scoring function assigns logits $p^i_{\curvearrowright}$ to each node $v_i \in V$, and the initial node is then sampled via the Gumbel-Softmax reparameterization:

$$\pi^i = \frac{\exp[(p^i_{\curvearrowright} + g_i)/\tau]}{\sum_j \exp[(p^j_{\curvearrowright} + g_j)/\tau]}, \tag{12}$$

where $\tau$ denote temperature, and $\pi^i$ parameterizes the initial-state distribution. On this basis, the initial state entropy can be formulated as:

$$H(\pi) = -\sum_i \pi^i \log \pi^i. \tag{13}$$

Intuitively, Eq. (13) biases the walk generation toward informative regions of the graph while retaining sufficient stochasticity for exploration.

**Step 2: Sequence-to-Subgraph Construction.** After initialization, the walk proceeds by repeatedly sampling the next node conditioned on the current state. Given node $v_i$ and its neighbors $\mathcal{N}(v_i)$, we define a policy head $f_\theta$ that produces unnormalized logits for each candidate edge:

$$p_{i\to j} = \begin{cases} f_\theta(z_{ij}), & Z \text{ is available,} \\ f_\theta(\text{concat}[x_i, x_j]), & \text{otherwise.} \end{cases} \tag{14}$$

Here, $z_{ij}$ and $x_i$, $x_j$ denote the attribute of edge $e_{ij}$ and node $v_i$, $v_j$, respectively. To restrict transitions to valid neighbors, we apply a masking operator:

$$\tilde{p}_{i\to j} = \begin{cases} p_{i\to j}, & e_{ij} \in E, \\ -\infty, & \text{otherwise.} \end{cases} \tag{15}$$

The masked logits are then transformed into a categorical transition distribution via temperature-controlled softmax:

$$\mathcal{T}_{ij} = \frac{\exp(\frac{\tilde{p}_{i\to j}}{\tau})}{\sum_{v_k \in \mathcal{N}(v_i)} \exp(\frac{\tilde{p}_{i\to k}}{\tau})}, \tag{16}$$

where $\tau$ is the temperature. On this basis, we adopt the Gumbel-Softmax reparameterization to repeatedly sample the next node while preserving gradient flow through $\tilde{p}_{i\to j}$.

Moreover, we add a small teleportation probability to the transition kernel, which reformulates:

$$T_{ij} = (1-\gamma) \cdot \mathcal{T}_{ij} + \gamma \cdot q_{i\curvearrowright}. \tag{17}$$

Here, $\gamma$ denotes the probability of teleportation and $q_{i\curvearrowright}$ uniformly links the current node $v_i$ to a random node on the graph[2]. Compared with Eq. (16), Eq. (17) offers two benefits: (i) it ensures the theoretical consistency of using $H(S)$ as a surrogate for the compressive target (see appendixes for more details); (ii) it promotes exploration and coverage of structurally diverse regions, which prevents the walk from being trapped in isolated areas, thereby improving the robustness of the learned representations.

**Step 3: Subgraph Aggregation and Prediction.** With each subgraph $G'_i$ constructed from the corresponding sequence $S_i$, we next proceed to merge $\{G'_i\}_{i=1}^k$ into a unified graph, which is then processed by subsequent layers. Specifically, we first derive: $\mathcal{G}_m = \bigcup_i^k G'_i$. The merged graph $\mathcal{G}_m$ contains the union of nodes from $\{G_i\}_{i=1}^k$ (without duplication), together with all traversed edges in $G$ among them. Next, $\mathcal{G}_m$ serves as the input to subsequent layers for the downstream task, where the message passing, readout, and prediction components can be flexibly instantiated. Notably, we operate on $\mathcal{G}_m$ rather than on individual subgraphs, as some structural patterns or discriminative signals can only be recognized from a global perspective, while isolating each subgraph might be overly biased or local (see Sec. 3.3 for further illustrations).

[2]Teleportation only modifies the local transition distribution by replacing $\mathcal{T}$ with the mixed kernel $T$, with the entropy terms evaluated directly under $T$.

Table 1: **Performance comparison** on the Benchmarking GNN Suite and Long-Range Graph Benchmarks. Best results in bold, second-best underlined, and baseline shaded.

| Datasets | MNIST | CIFAR10 | PATTERN | CLUSTER | ZINC | VOC-SP | COCO-SP | Pep-F | Pep-S |
|---|---|---|---|---|---|---|---|---|---|
| Num. # graph | 70K | 60K | 14K | 12K | 12K | 11.4K | 123.3K | 15.5K | 15.5K |
| AVG. # nodes | 70.6 | 117.6 | 118.9 | 117.2 | 23.2 | 479.4 | 476.9 | 150.9 | 150.9 |
| AVG. # edges | 564.5 | 941.1 | 3039.3 | 2150.9 | 24.9 | 2710.5 | 2693.7 | 307.3 | 307.3 |
| $\mathcal{L}_{\text{task}}$ | CE | CE | WCE | WCE | L1 | WCE | WCE | MCE | L1 |
| Metric | ACC ↑ | ACC ↑ | ACC ↑ | ACC ↑ | MAE ↓ | F1 ↑ | F1 ↑ | AP ↑ | MAE ↓ |
| GCN | $90.70_{\pm 0.22}$ | $55.71_{\pm 0.38}$ | $71.89_{\pm 0.33}$ | $68.50_{\pm 0.98}$ | $0.37_{\pm 0.01}$ | $0.208_{\pm 0.003}$ | $0.134_{\pm 0.001}$ | $68.60_{\pm 0.5}$ | $0.246_{\pm 0.001}$ |
| GIN | $96.48_{\pm 0.25}$ | $55.25_{\pm 1.53}$ | $85.39_{\pm 0.14}$ | $64.72_{\pm 1.55}$ | $0.53_{\pm 0.05}$ | $0.272_{\pm 0.005}$ | $0.213_{\pm 0.001}$ | $66.21_{\pm 0.7}$ | $0.247_{\pm 0.002}$ |
| GatedGCN | $97.34_{\pm 0.14}$ | $67.31_{\pm 0.31}$ | $85.57_{\pm 0.09}$ | $73.84_{\pm 0.33}$ | $0.28_{\pm 0.02}$ | $0.388_{\pm 0.004}$ | $0.292_{\pm 0.002}$ | $67.65_{\pm 0.5}$ | $0.248_{\pm 0.001}$ |
| GPS | $98.05_{\pm 0.13}$ | $72.30_{\pm 0.36}$ | $86.68_{\pm 0.06}$ | $78.02_{\pm 0.18}$ | $0.07_{\pm 0.01}$ | $0.375_{\pm 0.011}$ | $0.341_{\pm 0.004}$ | $65.35_{\pm 0.4}$ | $0.250_{\pm 0.001}$ |
| GRIT | $98.11_{\pm 0.11}$ | $76.47_{\pm 0.88}$ | $87.19_{\pm 0.08}$ | $80.02_{\pm 0.28}$ | $0.06_{\pm 0.01}$ | – | – | $69.88_{\pm 0.8}$ | $0.246_{\pm 0.001}$ |
| GRED | $98.83_{\pm 0.01}$ | $76.85_{\pm 0.19}$ | $86.76_{\pm 0.02}$ | $78.49_{\pm 0.10}$ | $0.08_{\pm 0.01}$ | – | – | $71.33_{\pm 0.1}$ | $0.246_{\pm 0.001}$ |
| Exphormer | $98.55_{\pm 0.03}$ | $74.69_{\pm 0.13}$ | $86.70_{\pm 0.03}$ | $78.07_{\pm 0.04}$ | – | $0.398_{\pm 0.004}$ | $0.346_{\pm 0.001}$ | $65.27_{\pm 0.4}$ | $0.248_{\pm 0.001}$ |
| GMN | $98.39_{\pm 0.21}$ | $75.76_{\pm 0.42}$ | $87.14_{\pm 0.12}$ | – | – | $0.439_{\pm 0.011}$ | $0.397_{\pm 0.010}$ | $70.71_{\pm 0.8}$ | $0.247_{\pm 0.003}$ |
| CRaWL | $97.94_{\pm 0.05}$ | $69.01_{\pm 0.26}$ | – | – | $0.09_{\pm 0.01}$ | – | – | $70.74_{\pm 0.3}$ | $0.251_{\pm 0.002}$ |
| **NWalker** | $98.69_{\pm 0.08}$ | $76.90_{\pm 0.46}$ | $86.97_{\pm 0.01}$ | $78.19_{\pm 0.19}$ | **0.05**$_{\pm 0.01}$ | $0.491_{\pm 0.004}$ | $0.440_{\pm 0.003}$ | $70.96_{\pm 0.8}$ | $0.246_{\pm 0.001}$ |
| **Ours** | **99.28**$_{\pm 0.03}$ | **80.16**$_{\pm 0.33}$ | **88.09**$_{\pm 0.03}$ | **80.38**$_{\pm 0.29}$ | $0.06_{\pm 0.12}$ | **0.511**$_{\pm 0.009}$ | **0.450**$_{\pm 0.002}$ | **72.43**$_{\pm 0.5}$ | **0.236**$_{\pm 0.001}$ |

In particular, the unified receptive field of $\mathcal{G}m$ integrates complementary structural signals from different walks, thereby mitigating local bias and revealing critical latent structures. Based on this formulation, the training objective is defined as:

$$\mathcal{L}_{train} = \beta \cdot H(S) + \mathcal{L}_{task} \tag{18}$$

Here, the task-specific objective $\mathcal{L}_{task}$ instantiates the sufficiency target[3] in Eq. (38), $H(S)$ accounts for the compression target, and $\beta$ is the Lagrangian multiplier.

# 3 Experiments

In this section, we show the proposed Entropic Walker can be flexibly adapted to various graph learning tasks, including molecular & protein function prediction, and image classification, *etc*.

## 3.1 Experimental Setup

### 3.1.1 Benchmarks and Evaluation Protocols

The Entropic Walker is evaluated on a diverse set of 15 benchmark datasets: (i) MNIST, CIFAR10, PATTERN, CLUSTER and ZINC from the Benchmarking GNN suite Dwivedi et al. (2023); (ii) PascalVOC-SP, COCO-SP, Peptides-func and Peptides-struct from the Long-Range Graph Benchmarks (LRGB) Dwivedi et al. (2022); (iii) OGBG-Molpcba, OGBG-PPA, and OGBG-Code2 from the Open Graph Benchmarks (OGB) Hu et al. (2020); (iv) Roman Empire (RE), Amazon Ratings (AR), MineSweeper (MS), Tolokers (TK) and Questions (QS) from the large graph benchmarks Platonov et al. (2023). Further dataset details along with the standard evaluation protocols are provided in Tab. 1-3, where ACC, MAE, AP, and RA denote accuracy, mean absolute error, average precision, and ROC AUC, respectively.

### 3.1.2 Implementation Details

In our implementation, the scoring function (for the initial state distribution) and the policy head (for transition probabilities) are both lightweight MLPs. For the message passing neural network (MPNN), we follow Chen et al. (2025); Wang et al. (2024a) and incorporate both local and global components. The local component employs a GIN-based operator for neighborhood aggregation, while the global component can be instantiated via attention layers Chen et al. (2022); Rampásek et al. (2022) or a virtual-node mechanism Tönshoff et al. (2024), depending on the dataset.

[3]For supervised classification, $\mathcal{L}_{task}$ is instantiated as the cross-entropy $\mathcal{L}_{task} = -\mathbb{E}[\log q_\phi(Y \mid G')]$. This objective can be viewed as a task-driven variational surrogate for the sufficiency term, since maximizing $\mathbb{E}[\log q_\phi(Y \mid G')]$ corresponds to maximizing a variational lower bound on $I(G'; Y)$ up to the constant $H(Y)$.

Table 2: **Performance** on OGB. Best results in bold, second-best underlined.

| Dataset | **MOLPCBA** | **PPA** | **ODE2** |
|---|---|---|---|
| Num. # graph | 437.9K | 158.1K | 452.7K |
| AVG. # nodes | 26.0 | 243.4 | 125.2 |
| AVG. # edges | 28.1 | 2266.1 | 124.2 |
| $\mathcal{L}_{\text{task}}$ | MCE | CE | CE |
| Metric | AP ↑ | ACC ↑ | F1 ↑ |
| SAT | – | $75.22_{\pm 0.56}$ | $0.194_{\pm 0.0028}$ |
| GIB† | $27.60_{\pm 0.33}$ | $71.94_{\pm 0.77}$ | $0.169_{\pm 0.0029}$ |
| GRDL† | $28.81_{\pm 0.30}$ | $72.78_{\pm 0.71}$ | $0.187_{\pm 0.0032}$ |
| GPS | $29.07_{\pm 0.28}$ | $\underline{80.15}_{\pm 0.33}$ | $0.189_{\pm 0.0024}$ |
| CRaWL | $29.86_{\pm 0.25}$ | – | – |
| NeuralWalker | $\underline{30.86}_{\pm 0.31}$ | $78.88_{\pm 0.59}$ | $\underline{0.196}_{\pm 0.0025}$ |
| Ours | **33.16**$_{\pm 0.29}$ | **80.52**$_{\pm 0.59}$ | **0.211**$_{\pm 0.0028}$ |

† Implemented in our codebase for fair comparison under a unified training and evaluation pipeline.

Table 3: **Performance** on large graphs. Best results in bold, second-best underlined.

| Dataset | **RE** | **AR** | **MS** | **TK** | **QS** |
|---|---|---|---|---|---|
| # nodes | 22,662 | 24,492 | 10,000 | 11,758 | 48,921 |
| # edges | 32,927 | 93,050 | 39,402 | 519,000 | 153,540 |
| $\mathcal{L}_{\text{task}}$ | NNL | NNL | NNL | NNL | NNL |
| Metric | ACC ↑ | ACC ↑ | AUC ↑ | AUC ↑ | AUC ↑ |
| GAT(-sep) | $88.75_{\pm 0.41}$ | $52.70_{\pm 0.62}$ | $93.91_{\pm 0.35}$ | $83.78_{\pm 0.43}$ | $76.79_{\pm 0.71}$ |
| GIB† | $84.69_{\pm 0.83}$ | – | $90.44_{\pm 0.79}$ | $80.60_{\pm 1.56}$ | – |
| GRDL† | $88.91_{\pm 1.22}$ | – | $91.60_{\pm 1.11}$ | $82.39_{\pm 1.21}$ | – |
| GPS | $82.00_{\pm 0.61}$ | $53.10_{\pm 0.42}$ | $90.63_{\pm 0.67}$ | $83.71_{\pm 0.48}$ | $71.73_{\pm 1.47}$ |
| Exphormer | $89.03_{\pm 0.37}$ | $53.51_{\pm 0.46}$ | $90.74_{\pm 0.53}$ | $83.77_{\pm 0.78}$ | $73.94_{\pm 1.06}$ |
| Polynormer | $92.55_{\pm 0.37}$ | $54.81_{\pm 0.49}$ | $97.46_{\pm 0.36}$ | $\underline{85.91}_{\pm 0.74}$ | $78.92_{\pm 0.89}$ |
| NeuralWalker | $\underline{92.92}_{\pm 0.36}$ | $54.58_{\pm 0.36}$ | $\underline{97.82}_{\pm 0.40}$ | $85.56_{\pm 0.74}$ | $78.52_{\pm 1.13}$ |
| Ours | **93.11**$_{\pm 0.66}$ | – | **98.94**$_{\pm 0.50}$ | **86.56**$_{\pm 0.82}$ | – |

† Implemented in our codebase for fair comparison under a unified training and evaluation pipeline.

Table 4: **Ablation study** on the contribution of each module in Entropic Walker.

| Setting | | | | **CIFAR** | **VOC-SP** | **OGB-M** | **PEP-S** |
|---|---|---|---|---|---|---|---|
| MPNN | RW | $H_{ini}$ | $H_{step}$ | ACC ↑ | F1 ↑ | AP ↑ | MAE ↓ |
| ✓ | – | – | – | 76.43 | 0.4443 | 28.31 | 0.2462 |
| ✓ | ✓ | – | – | 76.72 | 0.4909 | 30.52 | 0.2488 |
| ✓ | ✓ | ✓ | – | 77.61 | 0.4924 | 30.41 | 0.2467 |
| ✓ | ✓ | – | ✓ | <u>78.52</u> | <u>0.4987</u> | <u>31.69</u> | <u>0.2372</u> |
| ✓ | ✓ | ✓ | ✓ | **80.16** | **0.511** | **33.16** | **0.236** |

Table 5: **Ablation study** on the contribution of each design choice of Entropic Walker.

| Setting | | | | **CIFAR** | **VOC-SP** | **OGB-M** | **PEP-S** |
|---|---|---|---|---|---|---|---|
| EdgeAttr | GIB | EW | $\mathcal{G}_m$ | ACC ↑ | F1 ↑ | AP ↑ | MAE ↓ |
| ✓ | ✓ | – | ✓ | 76.07 | 0.445 | 28.41 | 0.2606 |
| ✓ | – | ✓ | – | 78.03 | 0.481 | <u>30.50</u> | <u>0.2461</u> |
| – | – | ✓ | ✓ | <u>79.07</u> | <u>0.493</u> | 29.41 | 0.2567 |
| ✓ | – | ✓ | ✓ | **80.16** | **0.511** | **33.16** | **0.2362** |

The task-specific objective $\mathcal{L}_{task}$ for each benchmark is listed in Tab. 1-3, where B-/W-/M-CE denote binary/weighted/multi-label cross-entropy, and NLL denotes negative log-likelihood. The teleportation probability $\gamma$ is fixed at 0.2 for all benchmarks, while the multiplier $\lambda$, length controller $L_{def}$, and other hyperparameters are provided in the appendixes.

All experiments are optimized by AdamW optimizer on 8 NVIDIA A6000 GPUs, where a linear warm-up increase of the learning rate at the beginning of the training followed by its cosine decay as in Chen et al. (2022; 2025).

### 3.2 Comparison with State-of-the-arts

We compare Entropic Walker against SOTA message passing GNNs, graph transformers, walk-based models, and GIB models. GNNs include GCN Kipf & Welling (2017), GIN Xu et al. (2019), and GatedGCN Bresson & Laurent (2017). Graph transformers using global message passing include SAT Chen et al. (2022), GPS Rampásek et al. (2022), GRIT Ma et al. (2023), GRED Ding et al. (2024c), GMN Behrouz & Hashemi (2024), Graph-Mamba Wang et al. (2024a), and Exphormer Shirzad et al. (2023). Walk-based models include CRaWL Tönshoff et al. (2023) and NeuralWalker Chen et al. (2025). GIB-based models incorporate the information bottleneck and include Wu et al. (2020); Wang & Fan (2024) Based on Tab. 1-3, we have the following observations:

(1) Entropic Walker achieves SOTA performance on most benchmarks, exhibiting consistent advantages over the cutting-edge graph learning models across diverse tasks.

(2) On LRGB, EW outperforms transformer-based models designed for long-range dependencies by up to 11.4%. This suggests that reducing structurally redundant or spurious connections can substantially improve learning on graphs with long-range dependencies.

(3) With controllable walk length and sampling budget, EW remains effective on large and complex graphs, outperforming most competitors on nearly all large-graph datasets.

(4) Performance is also influenced by the availability of edge attributes. On datasets such as ZINC and CLUSTER, where intrinsic edge attributes are unavailable, edge features are constructed by

concatenating node features. This may introduce duplicated information already present in the nodes, which can weaken the benefit of compression and partially limit performance (see Sec. 3.3).

### 3.3 Ablation Study

As reported in Tab. 4 & 5, we evaluate each component of our approach on various tasks. For simplicity, PascalVOC-SP, OGBG-Molpcba, and Peptides-struct are abbreviated as VOC-SP, OGB-M, and PEP-S, respectively.

We first clarify different settings in Tab. 4. "MPNN" and "RW" denote the message passing network and random walk, while $H_{ini}$ and $H_{step}$ denote the initial-state and step-wise entropy.

**Ablation on random walk.** Random walks generally benefit graph learning, with evident gains on image classification but relatively limited improvements on structural-property prediction tasks such as Peptides (see the first two rows of Tab. 4).
**Ablation on initial state entropy.** Using only $H_{ini}$ yields notable gain on vision tasks. We conjecture this is because the non-predictive information, such as background pixels, can be locally identified at the node level, whereas molecular and protein structures involve non-local, long-range interactions that cannot be captured by examining individual nodes alone.
**Ablation on step-wise entropy and EW.** $H_{step}$ and its combination with $H_{ini}$ constitute a leap in performance, suggesting that Entropic Walker effectively prunes redundant signals while preserving critical structures, thereby facilitating predictive information propagation.

We further investigate three additional factors: (1) a GIB-based alternative Wu et al. (2020) that explicitly accounts for the non-IID nature of graph structures; (2) the subgraph merging operation $\mathcal{G}_m$; and (3) the availability of edge attributes ("EdgeAttr). Results are reported in Tab. 5.

**Ablation on alternative GIB strategy.** The first row in Tab. 5 is obtained by replacing Step 1 & 2 in Fig. 1 with GIB, which iteratively samples ego graphs for each node. However, such local sampling may neglect non-local cues and introduce duplicated information across overlapping ego graphs, limiting its ability to learn compact representations. Thus it provides limited gains, especially on graph-level tasks such as OGBG-Molpcba and Peptides-struct.
**Ablation on subgraph merging.** Comparing the $2^{nd}$ and $4^{th}$ rows in Tab. 5, individually processing each subgraph seriously hinders the message passing process, resulting in restricted and biased perspectives and thus incurs decline.
**Ablation on edge attribute.** As discussed in Sec. 3.2, the absence of edge attributes degrades the performance. To verify this, we construct the edge features by concatenating node features, which essentially replicates existing content and results in consistently inferior performance across all benchmarks (comparing the $3^{rd}$ and $4^{th}$ rows in Tab. 5), in line with preceding analysis.

### 3.4 Understanding Entropic Walker: Entropy, Compactness, and Performance

We vary the entropy coefficient as $\beta' = r\beta$, with $r \in \{0, 0.5, 1, 2, 4\}$, and examine the resulting changes in walk entropy, structural retention, and predictive performance. Specifically, walk entropy is estimated as $H(S) = H_{ini} + \sum_{t=1}^{L-1} \frac{1}{K_t} \sum_{k=1}^{K_t} -\log \pi(e_t^{(k)} | v_{t-1}^{(k)})$, which provides a Monte Carlo estimate of Eq.(9) over $K_t$ sampled walks. For the node-level benchmarks PascalVOC-SP and PATTERN, all nodes are retained and structural compactness is therefore measured by edge retention. Based on Tab. 6, we show the followin observations:

(1) Increasing $\beta$ consistently reduces both walk entropy and structural retention across all datasets. On PascalVOC-SP, $r = 0 \rightarrow 4$ reduces $H(S)$ by 120.85 and edge retention by 42.31%, implying that lower walk entropy corresponds to more compact graph structures.

(2) For $r \in [0, 1]$, structural retention decreases while predictive performance consistently improves. On CIFAR, edge retention decreases by 37.72% while accuracy improves by 3.65%, suggesting that EW preferentially removes task-irrelevant structures while preserving predictive information.

(3) Beyond the optimal region, stronger compression does not immediately degrade predictive performance. At $r = 2$, EW retains substantially less structure while remaining comparable to or better than the unregularized model across all four datasets, suggesting that the task objective counterbalances compression by preserving predictive components.

Table 6: **Effect of entropy regularization on compactness and downstream performance.** † denotes the default setting ($r = 1$). Node and edge retention ratios (Node ret. and Edge ret.) are computed as $|V(G_m)|/|V(G)|$ and $|E(G_m)|/|E(G)|$, respectively, where $G_m$ denotes the retained subgraph. For node-level tasks, only Edge Ret. is reported since all nodes are preserved. Results are reported as mean ± standard deviation over ten independent runs.

| Dataset | Metric | $r = 0$ | $r = 0.5$ | $r = 1^{\dagger}$ | $r = 2$ | $r = 4$ |
|---|---|---|---|---|---|---|
| **CIFAR10** Graph-level | $H_{\text{start}}$ | $0.33_{\pm 0.02}$ | $0.16_{\pm 0.01}$ | $\approx 6.3\text{e}^{-2}$ | $\approx 4.5\text{e}^{-4}$ | $\approx 5.0\text{e}^{-5}$ |
| | $\bar{H}_{\text{step}}$ | $1.94_{\pm 0.03}$ | $1.69_{\pm 0.03}$ | $1.14_{\pm 0.04}$ | $0.78_{\pm 0.05}$ | $0.31_{\pm 0.05}$ |
| | $H(S)$ | $95.39_{\pm 1.23}$ | $82.97_{\pm 1.47}$ | $55.92_{\pm 1.96}$ | $38.22_{\pm 2.21}$ | $15.19_{\pm 2.45}$ |
| | Node ret. (%) | $98.10_{\pm 0.12}$ | $86.38_{\pm 0.35}$ | $73.61_{\pm 0.55}$ | $65.36_{\pm 0.75}$ | $59.02_{\pm 1.05}$ |
| | Edge ret. (%) | $94.23_{\pm 0.18}$ | $73.49_{\pm 0.45}$ | $56.51_{\pm 0.70}$ | $44.77_{\pm 0.90}$ | $38.33_{\pm 1.20}$ |
| | Acc. | $76.51_{\pm 0.30}$ | $78.52_{\pm 0.28}$ | $\mathbf{80.16}_{\pm 0.33}$ | $78.02_{\pm 0.40}$ | $73.16_{\pm 0.75}$ |
| **MolPCBA** Graph-level | $H_{\text{start}}$ | $2.17_{\pm 0.06}$ | $1.70_{\pm 0.07}$ | $1.22_{\pm 0.07}$ | $0.69_{\pm 0.07}$ | $0.11_{\pm 0.03}$ |
| | $\bar{H}_{\text{step}}$ | $0.89_{\pm 0.01}$ | $0.74_{\pm 0.02}$ | $0.58_{\pm 0.02}$ | $0.33_{\pm 0.03}$ | $0.14_{\pm 0.03}$ |
| | $H(S)$ | $23.53_{\pm 0.29}$ | $19.46_{\pm 0.37}$ | $15.14_{\pm 0.49}$ | $8.61_{\pm 0.60}$ | $3.47_{\pm 0.60}$ |
| | Node ret. (%) | $97.92_{\pm 0.12}$ | $89.47_{\pm 0.30}$ | $78.55_{\pm 0.50}$ | $69.77_{\pm 0.70}$ | $59.09_{\pm 0.95}$ |
| | Edge ret. (%) | $96.51_{\pm 0.15}$ | $84.80_{\pm 0.35}$ | $68.31_{\pm 0.55}$ | $58.01_{\pm 0.75}$ | $53.61_{\pm 1.00}$ |
| | AP | $30.62_{\pm 0.38}$ | $32.53_{\pm 0.32}$ | $\mathbf{33.16}_{\pm 0.28}$ | $32.08_{\pm 0.40}$ | $28.91_{\pm 0.65}$ |
| **VOC-SP** Node-level | $H_{\text{start}}$ | $0.18_{\pm 0.02}$ | $0.11_{\pm 0.01}$ | $0.07_{\pm 0.01}$ | $0.02_{\pm 0.01}$ | $\approx 4.5\text{e}^{-3}$ |
| | $\bar{H}_{\text{step}}$ | $1.69_{\pm 0.02}$ | $1.52_{\pm 0.03}$ | $1.31_{\pm 0.04}$ | $0.97_{\pm 0.05}$ | $0.63_{\pm 0.06}$ |
| | $H(S)$ | $167.49_{\pm 1.98}$ | $150.59_{\pm 2.77}$ | $129.76_{\pm 3.47}$ | $96.05_{\pm 4.46}$ | $46.64_{\pm 5.45}$ |
| | Edge ret. (%) | $98.21_{\pm 0.15}$ | $85.83_{\pm 0.40}$ | $69.16_{\pm 0.60}$ | $62.41_{\pm 0.85}$ | $55.90_{\pm 1.15}$ |
| | F1 | $48.32_{\pm 0.005}$ | $49.16_{\pm 0.005}$ | $\mathbf{51.11}_{\pm 0.009}$ | $48.49_{\pm 0.006}$ | $42.44_{\pm 0.090}$ |
| **PATTERN** Node-level | $H_{\text{start}}$ | $3.98_{\pm 0.08}$ | $3.25_{\pm 0.10}$ | $2.49_{\pm 0.12}$ | $1.12_{\pm 0.14}$ | $0.16_{\pm 0.06}$ |
| | $\bar{H}_{\text{step}}$ | $3.74_{\pm 0.02}$ | $3.16_{\pm 0.03}$ | $2.68_{\pm 0.04}$ | $2.03_{\pm 0.05}$ | $1.13_{\pm 0.06}$ |
| | $H(S)$ | $374.24_{\pm 1.78}$ | $316.09_{\pm 2.48}$ | $267.81_{\pm 3.47}$ | $202.09_{\pm 4.46}$ | $112.03_{\pm 5.94}$ |
| | Edge ret. (%) | $57.60_{\pm 0.25}$ | $45.94_{\pm 0.40}$ | $34.71_{\pm 0.55}$ | $27.69_{\pm 0.75}$ | $21.22_{\pm 1.00}$ |
| | Acc. | $86.81_{\pm 0.01}$ | $87.33_{\pm 0.06}$ | $\mathbf{88.09}_{\pm 0.03}$ | $87.76_{\pm 0.24}$ | $82.40_{\pm 0.65}$ |

(4) With overly large $\beta$, continued reductions in $H(S)$ and structural complexity eventually degrade predictive performance, as excessive compression may remove informative structures. This is consistent with our theoretical analysis, where reduced walk diversity can concentrate traversal on a small set of nearly deterministic paths and limit model expressiveness.

Overall, these results provide empirical evidence that walk entropy is a meaningful proxy for graph structural compactness. Combined with the task objective, EW reduces unnecessary structures while preserving predictive performance, consistent with the coding perspective that concentrating transition probabilities on fewer traversal patterns reduces coding uncertainty and yields more concise graph representations.

**Additional analysis.** Beyond the above analyses, we provide additional experiments on scalability and computational cost, performance under matched structural budgets, and sensitivity to the maximum walk length and teleportation probability. Detailed results are provided in Appendix.

## 4 Conclusion

In this work, we introduced Entropic Walker (EW), a principled framework for compact and predictive graph learning from an information-theoretic perspective. By characterizing structural complexity through transition entropy, EW provides a tractable proxy for structural redundancy and eliminates the need for mutual-information estimation. Implemented via a random-walk mechanism, it effectively removes superfluous structures while preserving sufficiency for downstream tasks. This work calls for rethinking how structural redundancy in graphs should be measured and opens promising directions for future research. For instance, extending the formulation to higher levels of the WeisfeilerLeman hierarchy or incorporating higher-order Markov dependencies.

## 5 AI USE STATEMETN

Generative AI tools were used solely to improve the readability of the manuscript and identify typographical errors. We have not used generative AI tools for research, ideation, methodology, or experiments. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES


Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *Proceedings of the International Conference on Learning Representations*, 2017.

Ali Behrouz and Farnoosh Hashemi. Graph mamba: Towards learning on graphs with state space models. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 119–130, 2024.

Ali Behrouz, Parsa Delavari, and Farnoosh Hashemi. Unsupervised representation learning of brain activity via bridging voxel activity and functional connectivity. In *Proceedings of the International Conference on Machine Learning*, 2024.

Karsten M. Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Proceedings of the International Conference on Data Mining*, pp. 74–81, 2005.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *CoRR*, abs/1711.07553, 2017.

Dexiong Chen, Laurent Jacob, and Julien Mairal. Convolutional kernel networks for graph-structured data. In *Proceedings of the International Conference on Machine Learning*, volume 119, pp. 1576–1586, 2020.

Dexiong Chen, Leslie O'Bray, and Karsten M. Borgwardt. Structure-aware transformer for graph representation learning. In *Proceedings of the International Conference on Machine Learning*, volume 162, pp. 3469–3489, 2022.

Dexiong Chen, Till Hendrik Schulz, and Karsten M. Borgwardt. Learning long range dependencies on graphs via random walks. In *Proceedings of the International Conference on Learning Representations*, 2025.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *Proceedings of the International Conference on Learning Representations*, 2018.

Shifei Ding, Wei Du, Ling Ding, Lili Guo, and Jian Zhang. Learning efficient and robust multi-agent communication via graph information bottleneck. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17346–17353, 2024a.

Shifei Ding, Wei Du, Ling Ding, Lili Guo, and Jian Zhang. Learning efficient and robust multi-agent communication via graph information bottleneck. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 17346–17353, 2024b.

Yuhui Ding, Antonio Orvieto, Bobby He, and Thomas Hofmann. Recurrent distance filtering for graph representation learning. In *Proceedings of the International Conference on Machine Learning*, 2024c.

Vijay Prakash Dwivedi, Ladislav Rampásek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Advances in Neural Information Processing Systems*, 2022.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *J. Mach. Learn. Res.*, 24:43:1–43:48, 2023.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *Proceedings of the World Wide Web Conference*, pp. 417–426, 2019.

Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks V. S. Lakshmanan, and Xuemin Lin. Efficient algorithms for densest subgraph discovery. *Proc. VLDB Endow.*, 12(11):1719–1732, 2019a.

Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks VS Lakshmanan, and Xuemin Lin. Efficient algorithms for densest subgraph discovery. *arXiv preprint arXiv:1906.00341*, 2019b.

Thomas Gärtner, Peter A. Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the International Conference on Computational Learning Theory*, volume 2777, pp. 129–143, 2003.

Floris Geerts. Walk message passing neural networks and second-order graph neural networks. *CoRR*, abs/2006.09499, 2020.

Aristides Gionis and Charalampos E. Tsourakakis. Dense subgraph discovery: KDD 2015 tutorial. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 2313–2314, 2015.

Van Thuy Hoang and O-Joun Lee. Pre-training graph neural networks on molecules by using subgraph-conditioned graph information bottleneck. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 17204–17213, 2025.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, 2020.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the International Conference on Machine Learning*, pp. 321–328, 2003.

Nikhil S Ketkar, Lawrence B Holder, and Diane J Cook. Subdue: Compression-based frequent pattern discovery in graph data. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pp. 71–76, 2005a.

Nikhil S Ketkar, Lawrence B Holder, and Diane J Cook. Subdue: Compression-based frequent pattern discovery in graph data. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pp. 71–76, 2005b.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*, 2017.

Katharina Limbeck, Lydia Mezrag, Guy Wolf, and Bastian Rieck. Geometry-aware edge pooling for graph neural networks. In *Advances in Neural Information Processing Systems*, 2025.

Jiayi Luo, Qingyun Sun, Beining Yang, Haonan Yuan, Xingcheng Fu, Yanbiao Ma, Jianxin Li, and Philip S. Yu. Robust graph condensation via classification complexity mitigation. In *Advances in Neural Information Processing Systems*, 2025.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip H. S. Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *Proceedings of the International Conference on Machine Learning*, volume 202, pp. 23321–23337, 2023.

Pierre Mahé, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. *J. Chem. Inf. Model.*, 45(4): 939–951, 2005.

Gaspard Michel, Giannis Nikolentzos, Johannes F. Lutzeyer, and Michalis Vazirgiannis. Path neural networks: Expressive and accurate graph neural networks. In *Proceedings of the International Conference on Machine Learning*, volume 202, pp. 24737–24755, 2023.

Giannis Nikolentzos and Michalis Vazirgiannis. Random walk graph neural networks. In *Advances in Neural Information Processing Systems*, 2020.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *Proceedings of the International Conference on Learning Representations*, 2023.

Ladislav Rampásek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, 2022.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *Proceedings of the International Conference on Learning Representations*, 2020.

Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 105(4):1118–1123, 2008.

Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *Proceedings of the International Conference on Machine Learning*, volume 202, pp. 31613–31632, 2023.

Naftali Tishby, Fernando C. N. Pereira, and William Bialek. The information bottleneck method. *CoRR*, physics/0004057, 2000.

Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing. *Trans. Mach. Learn. Res.*, 2023, 2023.

Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *Trans. Mach. Learn. Res.*, 2024, 2024.

Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. volume abs/2402.00789, 2024a.

Yan Wang, Wenju Hou, Nan Sheng, Ziqi Zhao, Jialin Liu, Lan Huang, and Juexin Wang. Graph pooling in graph neural networks: methods and their applications in omics studies. *Artificial Intelligence Review*, 57 (11):294, 2024b.

Zixiao Wang and Jicong Fan. Graph classification via reference distribution learning: Theory and practice. In *Advances in Neural Information Processing Systems*, 2024.

Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. In *Advances in Neural Information Processing Systems*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations*, 2019.

Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the International Conference on Data Mining*, pp. 721–724, 2002.

Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. In *Proceedings of the International Conference on Learning Representations*, 2021.

Junchi Yu, Jie Cao, and Ran He. Improving subgraph recognition with variational graph information bottleneck. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19374–19383, 2022.

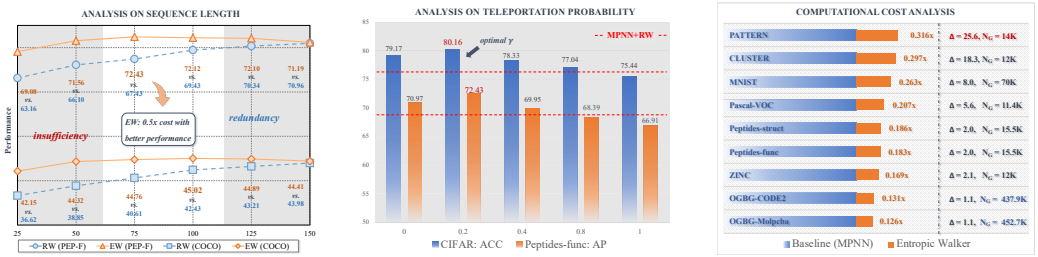
Ruiwen Yuan, Yongqiang Tang, Yanghao Xiao, and Wensheng Zhang. IBCS: learning information bottleneck-constrained denoised causal subgraph for graph classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 47 (3):1627–1643, 2025.

Mohammed J Zaki. Efficiently mining frequent embedded unordered trees. *Fundamenta Informaticae*, 66(1-2): 33–52, 2005.

Guibin Zhang, Xiangguo Sun, Yanwei Yue, Chonghe Jiang, Kun Wang, Tianlong Chen, and Shirui Pan. Graph sparsification via mixture of graphs. In *International Conference on Learning Representations*, 2025a.

Shuai Zhang, Junfeng Fang, Xuqiang Li, Hongxin Xiang, Alan Xia, Ye Wei, Wenjie Du, and Yang Wang. Iterative substructure extraction for molecular relational learning with interactive graph information bottleneck. In *Proceedings of the International Conference on Learning Representations*, 2025b.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *Proceedings of the International Conference on Machine Learning*, volume 119, pp. 11458–11468, 2020.




(a) Walk Length (b) Teleportation Probability (c) Computational Cost

Figure 2: (a): Evaluation on different walking lengths of random walker (RW) and the proposed Entropic Walker (EW). (b) Performance of our approach when adopting different teleportation probability $\gamma$. (c): Analysis of the extra computation brought by EW.

### 5.1 Qualitative Results

The qualitative results are summarized in Fig. 2a-2c, where "EW" in Fig. 2a indicates our framework shown in Fig. 1, "RW" replaces the entropic walker with the cutting-edge random walker Chen et al. (2025). $\Delta$ and $N_G$ denote the ratio of the number of edges to the number of nodes (*i.e.*, $|E|/|V|$) and the total of graphs, respectively.

**Analysis on walk length.** Observations of Fig. 2a are as follows: (i) Compared with the SOTA random walker Chen et al. (2025), our approach achieves both higher efficiency and accuracy. (ii) The performance of EW first climbs with the walk length, then stabilizes, and eventually declines. We deduce that such process illustrates a transition from insufficiency to minimal sufficiency and, finally, to redundancy. (iii) The performance of the baseline random walk grows sublinearly with complexity and soon saturates, reflecting the similar behavior as (ii) but with higher complexity and inferior results, further highlighting the importance of identifying minimal sufficient substructures. Additional sensitivity analysis on maximum walk length is provided in Tab. 9, showing that moderate walk lengths provide consistent gains while overly short or long walks may limit performance.

**Analysis on teleportation.** As illustrated in Fig. 2b, an appropriate teleportation probability benefits EW by mitigating sequence degeneracy and encouraging exploration. However, as $\gamma$ increases beyond 0.4, it becomes detrimentalc inducing excessive randomness that disrupts locality and structural consistency—thereby incurring severe decline, even below the random walk baseline. We further investigate the robustness of EW to different teleportation probabilities in Tab. 9, where similar trends are observed across substantially different datasets.

**Complexity analysis.** As shown Fig. 2c, the computational cost brought by our entropic walker is negligible for most benchmarks, and becomes relatively noticeable on small-scale benchmarks with extremely high edge density. We further provide performance—memory—runtime comparisons on representative large-scale graphs in Tab. 8 to evaluate the scalability of EW.

**Visualization.** As illustrated in Fig. 3, we visualize the learned subgraph of showcase in Peptides-struct benchmark using *networkx.spring_layout*, where colors refer to sampled paths. Observations are as follows: (i) EW effectively preserves the essential structures while filtering out considerable insignificant details, yielding a compact yet expressive structural representation. (ii) EW captures structurally diverse patterns across the graph, ensuring balanced attention between densely connected and peripheral regions. (iii) The teleportation mechanism enables escape from local minima, exhibiting benefits that complement its theoretical guarantee of reducing sequence degeneracy.

### 5.2 Analysis of Performance Gains under Matched Structural Budgets

We further conduct two complementary experiments to disentangle the effect of entropy-guided structural selection from additional model components and graph size reduction. Specifically, for node-level tasks, we construct matched baselines by randomly removing edges to match EWs edge retention ratios. For graph-level tasks, we randomly remove nodes according to EWs node retention ratios and retain valid remaining edges to obtain matched graphs. All variants use the same backbone, data split, and optimization settings. Results are reported in Tab. 7. Key observations are as follows:

(1) Without structural compression, EW and the corresponding baseline differ by at most 0.13% across all four datasets. This indicates that the additional walk-policy component alone cannot explain the observed gains.

(2) When structural compression is enabled, EW consistently improves performance, while the matched baselines suffer substantial degradation under random compression. This suggests that the performance improve-

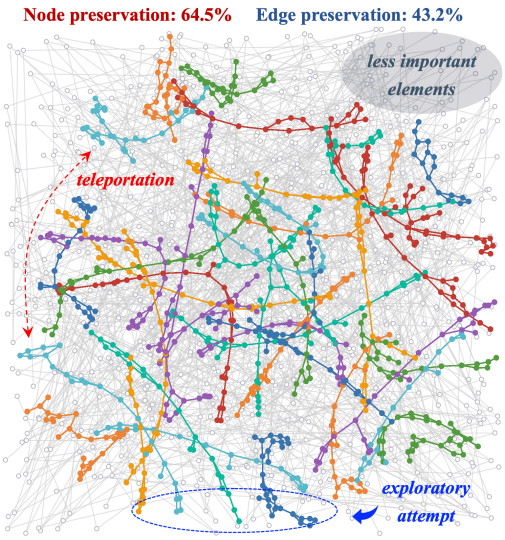



Figure 3: Visualization of learned walks on the full graph. Discontinuities indicate overlapping walks with shared nodes or edges shown in one color.

Table 7: **Performance comparison under full-graph and matched structural-budget settings.** EW and the baseline use identical configurations in the full-graph setting, while the matched-compression setting controls for retained graph structures. $\Delta$ denotes the performance gain of EW over the matched baseline.

| | Full graph | | Matched compression | | | Retained structure | |
|---|---|---|---|---|---|---|---|
| **Dataset (metric)** | **EW** | **Baseline** | **EW** | **Matched baseline** | $\Delta$ | $\frac{\lvert V(G_M)\rvert}{\lvert V(G)\rvert}$ | $\frac{\lvert E(G_M)\rvert}{\lvert E(G)\rvert}$ |
| PascalVOC-SP (F1) | 48.32 | 48.19 | **51.11** | 42.84 | **+8.27** | 100.00% | 69.16% |
| CIFAR10 (Accuracy) | 76.51 | 76.49 | **80.16** | 70.72 | **+9.44** | 73.61% | 56.51% |
| PATTERN (Accuracy-SBM) | 86.81 | 86.82 | **88.09** | 82.09 | **+6.00** | 100.00% | 34.71% |
| OGBG-MolPCBA (AP) | 30.62 | 30.55 | **33.16** | 22.16 | **+11.00** | 78.55% | 68.31% |

ment comes from entropy-guided structural selection rather than compression itself: EW can identify less informative structures to remove while retaining task-relevant information.

Together with Tab. 6, these results provide further evidence that graph structures often contain redundant or task-irrelevant components beyond those required for prediction. Effectively identifying and removing such components while preserving essential structures can reduce the influence of nuisance signals and yield more informative graph representations. These findings support the design intuition of EW: entropy-guided structural selection provides a practical mechanism for learning compact yet predictive graph representations.

### 5.3 Scalability Analysis on Large Graphs

To further demonstrate the scalability of EW, we conduct performance and efficiency comparisons on five representative large-graph benchmarks under identical hardware and batch settings. As shown in Tab. 8, EW consistently improves task performance by 1.51%-3.44 % with moderate overhead (3.6%-17.6% additional peak GPU memory and 3.8%-25.4% runtime increase), showing that EW remains applicable to large graphs with limited additional cost.

### 5.4 Additional Sensitivity Analysis

EW mainly involves three key hyperparameters: the entropy coefficient $\beta$, teleportation probability $\gamma$, and maximum walk length $L$. The impact of the entropy coefficient has been analyzed in Tab. 6, through the entropy-compactness-performance study. Here, we further examine the sensitivity of teleportation probability and maximum walk length across four representative benchmarks covering graph- and node-level tasks, differ-

Table 8: **Performance–efficiency comparison on large-graph benchmarks.** $\Delta$ denotes the absolute performance gain of EW over the baseline in percentage points. Memory and runtime are measured under identical hardware and software settings.

| Dataset | Metric | Performance ↑ | | | Memory (MB) ↓ | | Runtime (s) ↓ | |
|---|---|---|---|---|---|---|---|---|
| | | Baseline | EW | $\Delta$ | Baseline | EW | Baseline | EW |
| Roman Empire | Accuracy | 91.60 | **93.11** | +1.51 | **22,266** | 26,177 | **0.99** | 1.24 |
| Minesweeper | ROC-AUC | 96.52 | **98.94** | +2.42 | **18,430** | 20,834 | **0.71** | 0.89 |
| Amazon Ratings | Accuracy | 53.96 | **55.68** | +1.72 | **41,223** | 42,954 | **2.11** | 2.48 |
| Questions | ROC-AUC | 78.13 | **79.60** | +1.47 | **49,168** | 52,071 | **2.21** | 2.58 |
| Tolokers | ROC-AUC | 84.49 | **86.56** | +2.07 | **55,916** | 57,931 | **2.68** | 3.16 |

Table 9: **Sensitivity to the teleportation coefficient $\gamma$ and the walk-length budget $L$.** All remaining hyperparameters are held fixed when varying one factor.

**(a) Sensitivity to the teleportation coefficient $\gamma$**

| Dataset (metric) | $\gamma = 0$ | $\gamma = 0.2$ | $\gamma = 0.4$ | $\gamma = 0.6$ | $\gamma = 1.0$ |
|---|---|---|---|---|---|
| CIFAR10 (Accuracy ↑) | 79.17 | **80.16** | 78.33 | 77.04 | 75.44 |
| Peptides-func (AP ↑) | 70.97 | **72.43** | 69.95 | 68.39 | 66.91 |
| OGBG-PPA (Accuracy ↑) | 78.14 | **80.52** | 79.75 | 78.01 | 74.16 |
| Roman Empire (Accuracy ↑) | 92.29 | **93.31** | 92.04 | 91.69 | 88.37 |

**(b) Sensitivity to the walk-length budget $L$**

| Dataset (metric) | $L = 25$ | $L = 50$ | $L = 75$ | $L = 100$ | $L = 125$ | $L = 150$ |
|---|---|---|---|---|---|---|
| Peptides-struct (MAE ↓) | 0.274 | 0.249 | **0.236** | 0.238 | 0.245 | 0.251 |
| OGBG-Code2 (F1 ↑) | 0.169 | 0.186 | 0.205 | **0.211** | 0.207 | 0.191 |
| PATTERN (Accuracy-SBM ↑) | 86.58 | 87.78 | **88.09** | 87.35 | 86.69 | 86.42 |
| COCO-SP (F1 score ↑) | 42.15 | 43.23 | **45.04** | 44.62 | 44.29 | 44.01 |

ent domains, and diverse graph scales. In all sweeps, the remaining architectural and optimization settings are kept fixed.

**Teleportation probability.** A similar optimal range is observed around $\gamma = 0.2$ across CIFAR10, Peptides-func, OGBG-PPA, and Roman-Empire. Removing teleportation slightly degrades performance due to limited exploration, whereas excessive teleportation weakens structural locality. In particular, $\gamma = 1$, corresponding to purely random jumps, leads to consistently inferior performance across all four benchmarks. These results suggest that a moderate teleportation probability provides a stable choice across different datasets.

**Maximum walk length.** Performance initially improves as the walk-length budget increases, reaches a broad optimum around $L \in \{75, 100\}$, and then gradually declines. The best-performing setting is $L = 75$ on Peptides-struct, PATTERN, and COCO, and $L = 100$ on OGBG-Code2. This trend is consistent across graph- and node-level tasks: short walks may provide insufficient structural context, while excessively long walks introduce additional traversal patterns without further improving prediction.

## 6 On Sequence Degeneracy

By definition, sequence degeneracy refers to the ambiguity of the walk sequence $S$ given its induced subgraph $G'$: multiple distinct walk paths may lead to the same subgraph, yielding $H(S|G') \neq 0$. Under the Entropic Walker formulation, we show that the probability of such degeneracy decays exponentially with the walk length $L$. Consequently, the induced conditional entropy term is practically negligible under the considered setting.

### 6.1 Setup and Notations

Given graph $G = \{V, E\}$, where $V$ and $E$ denote the set of nodes and edges. A walk of length $L$ is generated according to a transition kernel:

$$T = (1-\gamma)\cdot\mathcal{T} + \gamma\cdot q_{\curvearrowright}, \tag{19}$$

where $\mathcal{T}$ is the transition matrix on $G$, $q_{\curvearrowright}$ uniformly links to a random node on $G$, and $\gamma$ denotes the teleportation probability. The resulting walk sequence is denoted by $S = \{v_1, ..., v_L\}$ and is mapped to an induced subgraph $G'$ via a deterministic function $\mathcal{F}(S) = G'$ by including all nodes in $S$ together with their traversed edges. A sequence $S_{\deg}$ is called degenerate *w.r.t* $G'$ if it yields the same induced subgraph.

Basically, the degenerate sequence $S_{\deg}$ is required to fully cover all elements of $G'$ while remaining strictly within its boundary. This allows us to upper-bound the degeneracy probability by the probability of staying entirely inside $G'$, multiplied by a factor that accounts for the number of distinct sequences realizing the same subgraph.

### 6.2 Bounding the Probability of Staying-Inside

For any node $i \in V(G')$, the probability of remaining inside $G'$ in a single step can be defined as:

$$P(v_{t+1} \in V(G') \mid v_t = i) = (1-\gamma)\sum_{j\in V(G')} T_{ij} + \gamma\frac{|V(G')|}{|V(G)|}. \tag{20}$$

On this basis, we define:

$$\lambda_i(G') \;=\; \sum_{j\in V(G')} \mathcal{T}_{ij}, \tag{21}$$

$$\lambda_{\max}(G') = \max_{i\in V(G')} \lambda_i(G'). \tag{22}$$

Here, $\lambda_{\max}(G')$ serves as an upper bound on the one-step retention probability within $G'$, and reformulates Eq. (20) as:

$$P(v_{t+1} \in V(G') \mid v_t = i) \le (1-\gamma)\lambda_{\max}(G') + \gamma\frac{|V(G')|}{|V(G)|}. \tag{23}$$

To characterize the minimal probability of leaving $G'$, we further define the leakage of $\mathcal{T}$ out of $G'$ as:

$$\delta_T(G') \;=\; \min_{i\in V(G')} \big(1-\lambda_i(G')\big) \;=\; \min_{i\in V(G')} \sum_{j\notin V(G')} \mathcal{T}_{ij}. \tag{24}$$

Apparently, we have:

$$\lambda_{\max}(G') \;\le\; 1-\delta_T(G'). \tag{25}$$

which further reformulates Eq. (23) into:

$$P(v_{t+1} \in V(G') \mid v_t = i) \tag{26}$$

$$\le (1-\gamma)\lambda_{\max}(G') + \gamma\frac{|V(G')|}{|V(G)|} \tag{27}$$

$$\le (1-\gamma)(1-\delta_T(g)) + \gamma\frac{|V(G')|}{|V(G)|} \tag{28}$$

$$= 1 - \underbrace{\Big[(1-\gamma)\delta_T(g) + \gamma\Big(1-\frac{|V(G')|}{|V(G)|}\Big)\Big]}_{:=\Delta(G')}. \tag{29}$$

In particular, Eq. (29) upper-bounds the one-step probability of remaining inside $G'$. In addition, $\Delta(G')$ can be specified as two components: (i) $(1-\gamma)\delta_T(g)$, leakage through the transition kernel $\mathcal{T}_{ij}$; (ii) $(1-\frac{|V(G')|}{|V(G)|})$ teleportation to nodes outside $G'$. Moreover, let $\pi(V(G')) = \sum_{i\in V(G')}\pi(i)$ denote the initial probability assigned to $V(G')$, we obtain the following bound on the probability that a walk of length $L$ stays entirely within $V(G')$.

**Lemma 1 (Stay-Inside Probability)** For any induced subgraph $G' \subsetneq G$ with $\delta_T(G') > 0$, the probability that an EW walk of length $L$ remains entirely inside $V(G')$ satisfies:

$$\Pr\big(S \subseteq V(G')\big) \;\le\; \pi(V(G'))\,\big(1-\Delta(G')\big)^{L-1}. \tag{30}$$

Furthermore, Eq. (30) directly yields an exponential form via the standard inequality:

$$\Pr(S \subseteq V(G')) \;\le\; \pi(V(G'))\,\exp\big(-\Delta(G')\,(L-1)\big), \tag{31}$$

which gives, for any $k \ge 0$:

$$\lim_{L\to\infty} L^k\,\Pr(S \subseteq V(G')) = 0. \tag{32}$$

In light of the preceding analysis, we conclude that stay-inside probability is negligible relative to polynomial-order terms. This establishes the first component in our bound on the sequence degeneracy probability.

### 6.3 Bounding Combinatorial Coverage Ratio

Beyond staying inside $G'$, a degenerate sequence $S_{deg}$ further requires a specific coverage pattern over $G'$. Consequently, within the set of all walks confined to $G'$, only a small combinatorial fraction can realize such a "cover-and-repeat configuration. In this section, we quantify this combinatorial contraction and show that it further reduces the degeneracy probabilityoften by more than ten orders of magnitude under typical settings.

Let $|V(G')| = m$, the total number of length-$L$ sequences drawn from $V(G')$ can be given by:

$$N_{\text{stay}}(L,m) = m^L. \tag{33}$$

Among these, the sequences that cover all nodes in $V(G')$ at least once are counted by:

$$N_{\text{cover}}(L,m) = m!\,\mathcal{S}(L,m) = \sum_{k=0}^{m}(-1)^k \binom{m}{k}(m-k)^L, \tag{34}$$

where $\mathcal{S}$ denotes Stirling partition number. On this basis, the fraction of length-$L$ sequences that achieve full coverage is measured by:

$$\rho(L,m) = \frac{N_{\text{cover}}(L,m)}{N_{\text{stay}}(L,m)} = \frac{m!\,S(L,m)}{m^L}. \tag{35}$$

Combining with the confinement bound in Eq. (30), we obtain the following relaxed upper bound on the degeneracy probability:

$$\Pr(S_{deg}) \;\lesssim\; \pi(V(G'))\,\big(1-\Delta(G')\big)^{L-1}\,\rho(L,m), \tag{36}$$

where the confinement factor $\big(1-\Delta(G')\big)^{L-1}$ captures the probability of staying inside $G'$, and $\rho(L,m)$ is the combinatorial coverage ratio. In particular, both terms decay at least exponentially in $L$: the confinement factor decays as $(1-\Delta(G'))^{L-1}$, while $\rho(L,m)$ decreases super-exponentially when $m$ is comparable to $L$, a regime that frequently appears in real-world graph learning benchmarks.

For instance, given $L \approx 100$ and $m \ge 85$ (a particularly common scenario in most benchmarks such as Peptides-struct/func, COCO, Pascal-VOC, OGBG-Code2, *etc.*), the degeneracy probability satisfies:

$$\Pr(S_{deg}) \;\lesssim\; \pi(V(G'))\,\big(1-\Delta(G')\big)^{99}\,\rho(100,85), \tag{37}$$

where $\rho(100,85) \approx 3.2\times10^{-23}$. Consequently, this yields an overall degeneracy probability that is empirically negligible for practical purposes.

## 7 Experimental Details

In this section, we provide additional dataset details and the hyperparameter settings used in our experiments. We follow the standard training protocols and the official train/validation/test splits for each benchmark.

### 7.1 Dataset Descriptions

**ZINC** Dwivedi et al. (2023). The ZINC dataset is a subset of the ZINC database and contains 12,000 molecular graphs representing commercially available chemical compounds. The task is to predict the constrained solubility (logP) in a regression setting. The dataset follows a standard split of 10,000/1,000/1,000 molecules for the training, validation, and test sets, respectively.

**MNIST and CIFAR10** Dwivedi et al. (2023). The MNIST and CIFAR10 benchmarks are derived from the original image datasets by segmenting each image into superpixels and connecting each superpixel to its eight nearest neighbors. Both datasets retain the original 10-class classification objective and follow the standard splits: 55K/5K/10K for MNIST and 45K/5K/10K for CIFAR10 (train/validation/test).

**COCO-SP and PascalVOC-SP Dwivedi et al. (2022)**. Similarly, both COCO and PascalVOC are derived from vision datasets by segmenting each image into superpixels. These superpixels are treated as nodes, leading to a node-classification task in which the objective is to predict the object class associated with each node.

Table 10: Hyperparameter settings for five benchmarks in the GNN benchmarking suite Dwivedi et al. (2023).

| HYPERPARAMETER | ZINC | MNIST | CIFAR10 | PATTERN | CLUSTER |
|---|---|---|---|---|---|
| Prediction Level | Graph | Graph | Graph | Node | Node |
| Local message passing | GIN | GIN | GIN | GIN | GIN |
| Global message passing | VN | None | None | VN | VN |
| Dropout | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Graph pooling | SUM | MEAN | MEAN | – | – |
| Batch size | 50 | 32 | 32 | 32 | 32 |
| Learning rate | 0.002 | 0.002 | 0.002 | 0.002 | 0.01 |
| Epochs | 500 | 100 | 100 | 100 | 100 |
| Warmup epochs | 50 | 5 | 5 | 5 | 5 |
| Weight decay | 0.0 | 1e-6 | 1e-6 | 0.0 | 0.0 |
| Lagrangian multiplier $\beta$ | 1e-4 | 2e-3 | 2e-3 | 5e-3 | 1e-4 |
| Length Controller $L_{def}$ | 50 | 50 | 50 | 75 | 150 |
| Sampling Rate | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |

Table 11: Hyperparameter settings for five benchmarks in the Long-Range Graph Benchmarks Dwivedi et al. (2022).

| HYPERPARAMETER | PascalVOC-SP | COCO-SP | Peptides-func | Peptides-struct |
|---|---|---|---|---|
| Prediction Level | Node | Node | Graph | Graph |
| Local message passing | GIN | GIN | GIN | GIN |
| Global message passing | Trans. | None | VN | VN |
| Dropout | 0.0 | 0.0 | 0.0 | 0.0 |
| Graph pooling | – | – | MEAN | MEAN |
| Batch size | 32 | 32 | 32 | 32 |
| Learning rate | 0.002 | 0.002 | 0.002 | 0.004 |
| Epochs | 200 | 200 | 200 | 200 |
| Warmup epochs | 10 | 10 | 10 | 10 |
| Weight decay | 1e-6 | 0 | 0 | 0 |
| Lagrangian multiplier $\beta$ | 1e-3 | 1e-3 | 1e-3 | 4e-3 |
| Length Controller $L_{def}$ | 75 | 75 | 75 | 75 |
| Sampling Rate | 0.5 | 0.5 | 0.5 | 0.5 |

**Peptides-func and Peptides-struct** Dwivedi et al. (2022). Both Peptides-func and Peptides-struct are derived from SATPdb by representing each peptide as an atomic graph. Peptides-func is formulated as a multi-label graph classification task, where the goal is to identify one or more functional classes of a peptide. In contrast, Peptides-struct is a graph regression benchmark that aims to predict continuous 3D structural properties of peptides.

**OGBG-Molpcba** Hu et al. (2020). This dataset is derived from PubChem BioAssay, where molecules are represented as graphs with atoms as nodes and chemical bonds as edges. The task is a multi-task graph classification problem aiming to predict the outcomes of 128 biological assays.

**OGBG-PPA** Hu et al. (2020). The protein-protein association (PPA) dataset contains graphs constructed from 1,581 species across 37 taxonomic groups. Each node represents a protein and each edge denotes a biologically meaningful association, equipped with 7-dimensional features. The task is 37-class graph classification, where each graph is assigned to its corresponding taxonomic group.

**OGBG-Code2** Hu et al. (2020). This dataset consists of abstract syntax trees (ASTs) extracted from source code in Python. Each graph represents a function, where nodes correspond to syntactic tokens and edges encode the AST structure. The prediction task is to generate the sequence of sub-tokens forming the functions name, making it a sequence-to-sequence graph prediction problem.

**Roman Empire** Platonov et al. (2023). This dataset is constructed from Wikipedia articles related to the Roman Empire. Each graph is a word co-occurrence network, where nodes represent unique words and edges indicate their local co-occurrence within a fixed window. The task is binary node classification, determining whether each word is contextually associated with the Roman Empire.

**Amazon Ratings** Platonov et al. (2023). This dataset is derived from the Amazon product co-purchasing data. Nodes and edges reflect products and whether they are frequently co-purchased. Each node is associated with a numerical user rating, and the task is to predict this rating based on the local and global structure of the co-purchasing graph.

Table 12: Hyperparameter settings for three benchmarks in the Open Graph Benchmark Hu et al. (2020).

| HYPERPARAMETER | OGBG-Molpcba | OGBG-PPA | OGBG-Code2 |
|---|---|---|---|
| Prediction Level | Graph | Graph | Graph |
| Local message passing | GatedGCN | GIN | GIN |
| Global message passing | VN | Performer | Trans. |
| Dropout | 0.4 | 0.4 | 0.0 |
| Graph pooling | MEAN | MEAN | MEAN |
| Batch size | 512 | 32 | 32 |
| Learning rate | 0.002 | 0.002 | 0.0003 |
| Epochs | 100 | 200 | 30 |
| Warmup epochs | 5 | 10 | 2 |
| Weight decay | 0 | 0 | 0 |
| Lagrangian multiplier $\beta$ | 3e-3 | 2e-3 | 2e-3 |
| Length Controller $L_{def}$ | 25 | 200 | 100 |
| Sampling Rate | 0.5 | 0.5 | 0.5 |

Table 13: Hyperparameter settings for the large graph benchmark Platonov et al. (2023).

| HYPERPARAMETER | Roman Empire | Amazon Ratings | Mine Sweeper | Tolokers | Questions |
|---|---|---|---|---|---|
| DROPOUT | 0.3 | 0.2 | 0.3 | 0.1 | 0.2 |
| Sampling Rate | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| Test Sampling Rate | 0.1 | 0.1 | 0.1 | 0.1 | 0.05 |
| $L_{def}$ | 800 | 800 | 800 | 800 | 800 |
| Learning Rate | 0.0005 | 0.0005 | 0.0005 | 0.001 | 5e-5 |

**Mine Sweeper** Platonov et al. (2023). This dataset transforms the classical Minesweeper game into a graph learning problem. Each node represents a grid cell, and edges connect spatially adjacent cells. Node labels correspond to the number of neighboring mines, making the task a node-level classification problem. The resulting graphs exhibit strong local structural patterns and require models to infer hidden spatial constraints from partially observed neighborhoods.

**Tolokers** Platonov et al. (2023). This dataset is constructed from crowdsourcing task interactions. Nodes represent worker accounts, and edges indicate co-participation or collaboration on the same tasks. The goal is node-level classification, where each node is labeled by a behavioral or participation category.

**Questions** Platonov et al. (2023). This dataset is derived from a question-answering platform, where each node corresponds to a question, and edges reflect semantic or topical similarity. The task is node classification, predicting the category of each question.

### 7.2 Hyperparameter Settings

Considering the large number of hyperparameters and datasets, we did not conduct exhaustive dataset-specific search beyond the ablation studies in Sec. 3.3. Instead, for each benchmark we basically follow the standard backbone architectures and training protocols used by the corresponding baselines. In practice, most hyperparameters reported in Tabs. 10-13 are inherited from the benchmark-specific baseline settings, while only a small number of EW-specific hyperparameters, mainly the entropy coefficient $\beta$ and the length controller $L_{def}$, are adjusted when necessary.

As illustrated in Tabs. 10-13, we summarize the key hyperparameters, architectural settings, and optimization details used throughout our experiments [4]. In particular, sampling rate denotes the ratio of initial nodes (*i.e.*, random walks) to the entire node set, and the coefficient of the entropy term $\beta$ is typically set around $10^{-3}$ for two reasons:

(i) **Avoiding path collapse.** A large $\beta$ usually leads to aggressive compression, causing paths to collapse certain ones. This severely reduces structural diversity and increases the risk of converging to suboptimal local minima.

(ii) **Matching numeric scales.** $H(S)$ aggregates entropy over all walk steps and is therefore numerically larger than the task loss. A small coefficient keeps the entropy term in the regularization regime, ensuring that it influences optimization without overwhelming the predictive objective.

[4]We report these dataset-level settings mainly for reproducibility. They should not be interpreted as exhaustive per-dataset tuning, since the vast majority of architectural and optimization choices are kept consistent with the corresponding benchmark baselines, and only the EW-specific regularization strength and walk-length control are lightly adjusted.

# 8 Related Work

## 8.1 Compact Graph Learning

Compact graph learning aims to retain task-relevant structures while removing superfluous or task-irrelevant components. Existing approaches range from heuristic substructure selection based on predefined graph statistics Fang et al. (2019b); Ketkar et al. (2005a) to data-driven graph compression methods such as sparsification, pooling, and condensation Zhang et al. (2025a); Wang et al. (2024b); Luo et al. (2025). While effective in practice, these methods often rely on task-specific heuristics and may discard structurally important information without a unified characterization of graph redundancy Limbeck et al. (2025).

Another line of research follows the information bottleneck (IB) principle Tishby et al. (2000), which can be written in the graph setting as:

$$\mathcal{L}_{GIB} = \underbrace{I(G;G')}_{\text{compression target}} - \beta \cdot \underbrace{I(G';Y)}_{\text{sufficiency target}}, \tag{38}$$

which intends to learn a subgraph $G'$ that preserves task-relevant information while removing superfluous structures. However, directly optimizing $I(G;G')$ on graph data remains challenging, since graph structures are high-dimensional, discrete, non-Euclidean, and highly interdependent, making mutual information difficult to define and estimate tractably.

In this work, we characterize structural complexity from the transition perspective, thereby avoiding direct mutual-information estimation and aligning the compactness objective more naturally with graph learning by decoupling structural complexity from subgraph diversity.

## 8.2 Random Walk for Graph Learning

Early graph learning studies leveraged random walk primarily as a measure of similarity for tasks such as community detection Rosvall & Bergstrom (2008), graph classification Borgwardt & Kriegel (2005). Typically, walk kernels were introduced to efficiently compare structural patterns across graphs Gärtner et al. (2003); Kashima et al. (2003), while non-backtracking walks were later explored for molecular graph analysis Mahé et al. (2005).

In deep graph learning, random walks have been employed as an efficient and structure-aware mechanism to enhance the expressivity of GNNs. A seminal work Nikolentzos & Vazirgiannis (2020) introduces a differentiable random-walk kernel and integrates it into a GNN architecture. Subsequent studies extend this idea from structural augmentation to sequential modeling. For instance, MPGNN Geerts (2020) incorporates random walks as auxiliary node features to develop the 2-dimensional WeisfeilerLehman framework, while GCKN Chen et al. (2020) aggregates walk-based path features into graph convolution layers. PathNN Michel et al. (2023) further utilizes paths emanating from nodes to update node embeddings, and RWGNN Nikolentzos & Vazirgiannis (2020) designs differentiable walk kernels for subgraph comparison via parameterized anchor graphs. Building on these advances, CRaWl Tönshoff et al. (2023) encodes subgraphs encountered along walks into sequential representations processed by 1D CNNs, whereas NeuralWalker Chen et al. (2025) models each walk as a sequence using a state-space formulation to better capture long-range dependencies, achieving superior performance over GNN and graph transformer baselines.

However, these models focus on enhancing the expressivity by including as much information as possible, and neglect to restrict task-irrelevant noise and redundancy. This not only degrades generalization ability, but further prohibits the model from identifying critical subgraphs or structures, which is vital for diverse graph learning tasks like biochemistry Zhang et al. (2025b) and causal subgraph inference Yuan et al. (2025). In contrast, our approach explicitly regularizes the information flow and effectively eliminates redundant signals while preserving the critical substructures required for prediction.