ON STRUCTURAL EXPRESSIVE POWER OF GRAPH TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

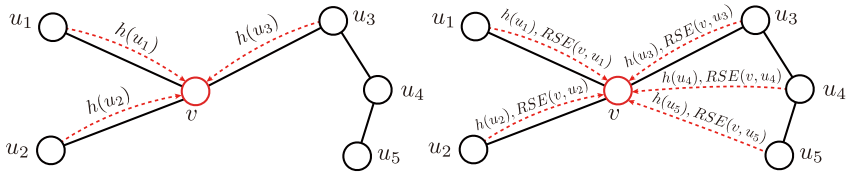
Graph Transformer (GT) has recently received wide attention in the research community with its outstanding performance, yet its structural expressive power has not been well analyzed. Inspired by the connections between Weisfeiler-Lehman (WL) graph isomorphism test and graph neural network (GNN), we introduce **GT test**, a generalized graph isomorphism test algorithm as a powerful theoretical tool for exploring the structural discriminative power of graph Transformers. We theoretically prove that the GT test is an expressivity upper bound on a wide range of graph Transformers, and the representational power of GT test can be approximated by a simple Transformer network arbitrarily under certain conditions. With the GT test, we show how graph Transformers' expressive power is determined by the design of structural encodings, and present conditions that make the expressivity of graph Transformers beyond WL test and GNNs. Moreover, motivated by the popular shortest path distance encoding, we follow the theory-oriented principles and develop a provably stronger structural encoding method, Shortest Path Induced Subgraph (*SPIS*) encoding. Our theoretical findings provide a novel and practical paradigm for investigating the expressive power of graph Transformers, and extensive synthetic and real-world experiments empirically verify the strengths of our proposed methods.

1 INTRODUCTION

In the last decade, graph neural network (GNN) (Kipf & Welling, 2016; Veličković et al., 2017) has become the prevalent neural architecture for deep learning on graph data. Following the message-passing scheme, GNNs learn the vector representation of node v by iteratively aggregating and transforming features of its neighborhood nodes. Recent studies (Xu et al., 2018) have proved that Weisfeiler-Lehman (WL) graph isomorphism test can measure the theoretical expressive power of message-passing GNNs in distinguishing graph structures (Weisfeiler & Leman, 1968).

While in the last few years, the Transformer architecture (Vaswani et al., 2017) has achieved broad success in various machine learning tasks. On graph representation learning, though with higher complexity than GNNs, recent works (Ying et al., 2021; Kreuzer et al., 2021) have proved that graph Transformers can successfully model large-scale graph data and deliver state-of-the-art performance on real-world benchmarks. However, despite advances in empirical benchmark results, the theoretical expressive power of graph Transformers has not been deeply explored. Compared with GNN's message-passing strategy, which only includes neighborhood aggregation, most graph Transformers represent nodes by considering all pair-wise interactions in the input graph, meaning that every node has a *global receptive field* at each layer. Besides, since vanilla self-attention is ignorant of node ordering, like positional encodings in language models, graph Transformers must design various *structural encodings* as a soft inductive bias to leverage graph structural information. Therefore, previous methods like WL test can no longer be used to analyze the expressivity of graph Transformers, considering the substantial differences between two model architectures. The natural questions arise: *How to characterize the structural expressive power of graph Transformers? How to build expressive graph Transformers that can outperform the WL test and GNNs?*

Our key to answering the questions above is **GT test**, a generalized graph isomorphism test algorithm designed to characterize the expressivity of graph Transformer (GT), as illustrated in Figure 1. Specifically, GT test represents a family of graph isomorphism test algorithms whose label update strategy is shaped by predefined *structural encodings*. For every input graph, GT test first inserts



(a) WL Test (b) GT Test

Figure 1: An illustration of the node label update strategies of WL test and GT test.

absolute structural encodings to the initial node labels. Then during each iteration, unlike WL test which updates the node label of v by hashing the multiset of its neighborhood node labels $\{\!\{h(u) : u \in \mathcal{N}(v)\}\!\}$, GT test hashes $\{\!\{(h(u), \text{RSE}(u, v)) : u \in V\}\!\}$, the collection of all node labels together with *relative structural encodings* to the central node. We theoretically prove that GT test is an expressivity upper bound on any graph neural model that learns structural information via structural encodings, including most graph Transformers (Theorem 1). Moreover, with the universal approximation theorem of Transformers (Yun et al., 2019), we show under certain assumptions, the expressivity of GT test can be approximated at any precision by a simple Transformer network which incorporates relative structural encodings as attention biases (Theorem 2). These conclusions guarantee that GT test can be a solid theoretical tool for our deeper investigation into the expressivity of graph Transformers.

Since the label update strategy of GT test is driven by structural encoding, we next develop general theories to understand the characteristics of structural encodings better. Our central result shows that one can compare the expressivity and convergence rate of GT tests by looking into the relationship between their structural encodings (Theorem 3), which provides us with a simple and powerful solution to analyze the representational capacity of GT test and graph Transformers. We show WL test can be viewed as a nested case of GT test (Theorem 4), and theoretically characterize how to design structural encodings that make graph Transformers more expressive than WL test and GNNs. We demonstrate that graph Transformers with the shortest path distance (*SPD*) structural encodings (like Graphormer (Ying et al., 2021)) are strictly more powerful than the WL test (Theorem 5), and they have distinctive expressive power that differs from encodings that focus on local information (Proposition 1). Based on *SPD* encodings, we follow the theoretical guidelines and design *SPIS*, a provably more powerful structural encoding (Theorem 6) with profound representational capabilities (Proposition 2-3). Our synthetic experiments verify that *SPIS* has remarkable expressive power in distinguishing graph structures, and the performances of existing graph Transformers can be consistently improved when equipped with the proposed *SPIS*.

Contributions. We summarize the main contributions of this work as follows:

- We introduce the GT test algorithm and prove it well characterizes the expressive power of various graph Transformers (Section 3, Theorem 1-2).
- Using the GT test, we develop a generalized theoretical framework on *structural encodings* that determines the expressivity of graph Transformers, and show how to make graph Transformers more expressive than WL test and GNNs (Section 4, Theorem 3-4).
- We conduct in-depth investigation into the expressivity of the existing *SPD* structural encoding, and propose a provably more powerful encoding method *SPIS* (Section 5, Theorem 5-6).
- Synthetic and real-world experiments demonstrate that *SPIS* has outstanding expressive power in distinguishing graph structures, and performances of benchmark graph Transformers are dominated by the theoretically more powerful *SPIS* encoding (Section 6).

Overall, we build a general theoretical framework for analyzing the expressive power of graph Transformers, and propose the *SPIS* structural encoding to push the boundaries of both expressivity and performance of graph Transformers. For clarity, one can find a detailed review of related works in Appendix B.

2 PRELIMINARIES

Basic Notations. Let $G = (V, E)$ be a graph where $V = \{v_1, v_2, \ldots, v_n\}$ is the node set that consists of n nodes, and $E \subset V \times V$ is edge set. Let $h_0 : V \to \mathcal{X}$ defines the input feature vector (or

label) attached to nodes, where $\mathcal{X} \subset \mathbb{R}^d$ is the feature space. In this paper, we only consider simple undirected graphs with node features, and we use $\mathcal{G}$ to denote the set of all possible labeled simple undirected graphs.

Structural Encodings. Generally, *structural encoding* is a function that encodes structural information in G to numerical vectors associated with nodes or node tuples of V. In the scope of this paper, we mainly use two types of structural encodings: *absolute structural encoding* (ASE), which represents absolute structural knowledge of individual nodes, and *relative structural encoding* (RSE), which represents the relative structural relationship between two nodes in the entire graph context. For a certain graph Transformer model, its structural encoding scheme consists of both absolute and relative encodings, and we present the formal definition below:

Definition 2.1 (Structural Encoding)**.** *A **structural encoding scheme** $S = (f_A, f_R)$ is a pair of functions, where for any graph $G = (V, E)$, $f_A(v, G) \in \mathcal{C}$ is the absolute structural encoding of any node $v \in V$, $f_R(v, u, G) \in \mathcal{C}$ is the relative structural encoding of any node pair $(v, u) \in V \times V$, and $\mathcal{C}$ is the target space. A structural encoding scheme is called **regular** if the relative structural encoding function satisfies $f_R(v, v, G) \neq f_R(v, u, G)$ for $u, v \in V$ and $u \neq v$.*

For example, we can use degree as an absolute structural encoding of a node, and use the shortest path distance between two nodes as the relative structural encoding of a node pair. We will discuss structural encodings more in the following sections.

Graph Transformers. Existing works Dwivedi & Bresson (2020); Ying et al. (2021); Kreuzer et al. (2021); Zhao et al. (2021) generally utilize the Transformer architecture to model graph data by leveraging different types of structural information via various modifications to the network. We mathematically formalize this crucial structural information as *structural encoding*. A detailed investigation into existing graph Transformers can be found in Appendix B, and in Appendix D we also show how their expressivity can be characterized with GT test.

3 GT Test and Graph Transformers

In this section, we mathematically formalize the GT test algorithm and theoretically prove that GT test well characterizes the expressive power of graph Transformers. Appendix A provides detailed proofs for all theorems and propositions in the following sections.

3.1 From WL Test to GT Test

Generally, previous GNN-based methods represent a node by summarizing and transforming its neighborhood information. This strategy leverages graph structure in a *hard-coded* way, where the structural knowledge is reflected by removing the information exchange between non-adjacent nodes. WL test is a high-level abstraction of this learning paradigm. However, graph Transformers take a fundamentally different way of learning graph representations. Without any hard inductive bias, self-attention represents a node by aggregating its semantic relation between every node in the graph, and structural encodings guide this aggregation as a *soft* inductive bias to reflect the graph structure. The proposed GT test then becomes a generalized algorithm for this powerful and flexible learning scheme by updating node labels based on the entire label set of nodes and their relative structural encoding to the central node, defined as follows:

Definition 3.1 (GT Test)**.** *Let the input be a labeled graph $G = (V, E)$ with label map $h_0 : V \to \mathcal{X}$. For structural encoding scheme $S = (f_A, f_R)$, its corresponding GT test algorithm first computes the initial label mapping $g_0 : V \to \mathcal{X}$ by adding the absolute structural encodings:*

$$g_0(v) = \Phi_0(h_0(v), f_A(v, G)), \tag{1}$$

where Φ_0 is a injective function that maps the tulple to $\mathcal{X}$. Then GT test iteratively updates node labels of G, where at the t-th iteration, the updated node label mapping $g_t : V \to \mathcal{X}$ is computed as

$$g_t(v) = \Phi\left(\{\!\{(g_{t-1}(u), f_R(v, u, G)) : u \in V\}\!\}\right), \tag{2}$$

where Φ is a function that injectively maps the collection of all possible multisets of tuples in the r.h.s. of Equation 2 to $\mathcal{X}$. We say two graphs G_1, G_2 are distinguished as non-isomorphic by S-GT test if after t iterations, S-GT generates $\{\!\{g_t(v) : v \in V_1\}\!\} \neq \{\!\{g_t(v) : v \in V_2\}\!\}$ for some t.

Note that for structural encoding scheme S we use S-GT to denote its corresponding GT test algorithm. Following its definition, we will show that GT test characterizes a wide range of graph neural models that leverage graph structure as a soft inductive bias:

Theorem 1. *For any structural encoding scheme $S = (f_A, f_R)$ and labeled graph $G = (V, E)$ with label map $h_0 : V \to \mathcal{X}$, if a graph neural model $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ satisfies the following conditions:*

1. *$\mathcal{A}$ computes the initial node embeddings with*

$$l_0(v) = \phi(h_0(v), f_A(v, G)), \tag{3}$$

2. *$\mathcal{A}$ aggregates and updates node embeddings iteratively with*

$$l_t(v) = \sigma(\{\!\{(l_{t-1}(u), f_R(v, u, G)) : u \in V\}\!\}), \tag{4}$$

   where ϕ and σ above are model-specific functions,

3. *The final graph embedding is computed by a global readout on the multiset of node features $\{\!\{l_t(v) : v \in V\}\!\}$.*

then for any labeled graphs G_1 and G_2, if $\mathcal{A}$ maps them to different embeddings, S-GT also decides G_1 and G_2 are not isomorphic.

In the GT test framework outlined by Theorem 1, Appendix D presents examples of characterizing the expressivity of existing graph Transformer models using certain structural encoding , including Dwivedi & Bresson (2020); Ying et al. (2021); Kreuzer et al. (2021); Zhao et al. (2021); Chen et al. (2022) . Notably, in Appendix A.1 we provide a more generalized version of Theorem 1 which proves that the widely adopted virtual node trick (Ying et al., 2021) has no influence on the maximum model expressive power.

3.2 Theoretically Powerful Graph Transformers

Though the maximum representational power of most graph Transformer models has been well characterized by GT test, it is still unknown if there exists a graph Transformer model that can reach its expressivity upper bound. Transformer layers are composed of self-attention module and feed-forward network, which drive them much more complex than standard GNN layers, making it challenging to analyze the expressive properties of graph Transformers. Thanks to the universal approximation theorem of Transformers (Yun et al., 2019), our next theoretical result demonstrates that under certain conditions, a simple graph Transformer model which leverages relative structural encodings as attention biases via learnable embedding layers (named as bias-GT) can arbitrarily approximate the GT test iterations for any structural encoding design:

Theorem 2. *For any regular structural encoding scheme S, graph order n, $1 < p < \infty$ and $\epsilon > 0$, let f_t represent the function of S-GT with t iterations. Then f_t can be approximated by a bias-GT network g with S such that $\mathsf{d}_p(f_t, g) < \epsilon$ if (i) the feature space $\mathcal{X}$ is compact, (ii) Φ can be extended to a continuous function with respect to node labels.*

In Theorem 2 we define f_t by stacking all labels generated by GT test with t iterations, and $\mathsf{d}_p(f_t, g)$ is the maximum ℓ^p distance between f_t and g when changing the input graph structure. The detailed descriptions for $f_t, g, \mathsf{d}_p, \Phi$ and the bias-GT network are provided in Appendix A.2.

Under certain conditions, Theorem 2 guarantees that the simple Transformer network bias-GT is theoretically capable of capturing structural knowledge introduced as attention biases and arbitrarily approximating its expressivity upper bound, though a good approximation may require many Transformer layers. Overall, considering that the simple bias-GT network (which can be viewed as a simplification of existing graph Transformers like Graphormer (Ying et al., 2021)) is one instance among the most theoretically powerful graph Transformers, one can translate the central problem of characterizing the expressive capacity of graph Transformers into understanding the expressivity of GT test, which is determined by the design of structural encodings.

4 General Discussions on GT Test and Structural Encodings

In this section, we develop a unified theoretical framework for analyzing structural encodings and the expressivity of GT test. One can tell that each GT test iteration has quadratic complexity with respect to the graph size and is more computationally expensive than WL, yet we will prove in the following text that GT test could exhibit extraordinary expressive power and lower necessary iterations when combined with a variety of structural encodings. We first present concrete examples and show how

the expressivity of structural encodings can be compared. Based on these findings, we prove that WL test is a nested case of GT test and theoretically characterize how to design structural encodings exceeding the expressivity of WL test. More discussions are provided in Appendix C.

4.1 Examples of Structural Encodings

Identical Encoding. The simplest encoding scheme assigns identical information to every node and non-duplicated node pair. Formally, let $\mathit{id} = (\mathit{id}_A, \mathit{id}_R)$ be the identical encoding scheme, then for $G = (V, E)$ and $v, u \in V$, $\mathit{id}_A(v, G) = 0, \mathit{id}_R(v, u) = 1, \mathit{id}_R(v, v) = 0$.

Node Degree Absolute Encoding. A common strategy for injecting absolute structural knowledge to node embeddings in the entire graph context is using the node degree as an additional signal. For graph $G = (V, E)$ and $v \in V$, let $\mathit{Deg}_A(v, G)$ be the degree of node v, then Deg_A is the node degree absolute encoding function.

Neighborhood Relative Encoding. Neighborhood relative encoding $\mathit{Neighbor}_R$ is a basic example that encodes edge connections. For $G = (V, E)$ and $v, u \in V$, it is defined as

$$\mathit{Neighbor}_R(v, u, G) = \begin{cases} 1, \text{ if } (v, u) \in E, \\ 2, \text{ if } (v, u) \notin E, \end{cases} \tag{5}$$

and $\mathit{Neighbor}_R(v, v, G) = 0$. We also use $\mathit{Neighbor} = (\mathit{id}_A, \mathit{Neighbor}_R)$ to denote the encoding scheme that combines $\mathit{Neighbor}_R$ with identical absolute encoding. Intuitively, we will show that *Neighbor* precisely shapes the expressivity of WL test.

Shortest Path Distance Relative Encoding. First introduced by (Ying et al., 2021), shortest path distance (SPD) is a popular choice for representing relative structural information between two nodes in the graph. We formulate it as

$$\mathit{SPD}_R(v, u, G) = \begin{cases} \text{the SPD between } v \text{ and } u \text{ in } G, & \text{if } v \text{ and } u \text{ are connected,} \\ \infty, & \text{if } v \text{ and } u \text{ are not connected,} \end{cases} \tag{6}$$

where ∞ can be viewed as an element in $\mathcal{C}$ and $\mathit{SPD}_R(v, v, G) = 0$. We also define the *SPD* structural encoding scheme as $\mathit{SPD} = (\mathit{id}_A, \mathit{SPD}_R)$.

4.2 Structural Encoding Determines the Expressiveness and Convergence Rate of GT test

Our next theoretical result is based on the intuitive idea that if one can infer the structural information in scheme S from another encoding scheme S', then S' should be generally more powerful and converge faster on graphs as it contains more information. To formulate this theoretical insight, we start by defining a partial ordering to characterize the relative discriminative power of structural encodings:

Definition 4.1 (Partial Order Relation on Structural Encodings)**.** *For two structural encoding schemes $S = (f_A, f_R)$ and $S' = (f'_A, f'_R)$, we call $S' \succeq S$ if there exist mappings p_A, p_R such that for any $G = (V, E)$ and $v, u \in V$ we have*

$$f_A(v, G) = p_A(f'_A(v, G)), \tag{7}$$

$$f_R(v, u, G) = p_R(f'_R(v, u, G)). \tag{8}$$

With the definition above, we next present the central theorem that shows structural encoding determines the expressiveness and convergence rate of GT test:

Theorem 3. *For two structural encoding schemes S and S', if $S' \succeq S$, then*

(1) S'-GT is more expressive than S-GT in testing non-isomorphic graphs.[1]

(2) for a pair of graphs G_1 and G_2 that S-GT distinguishes as non-isomorphic after t iterations, S'-GT can distinguish G_1 and G_2 as non-isomorphic within t iterations.

[1]For two isomorphic testing algorithms A and B, we say A is more expressive than B if any non-isomorphic graphs distinguishable by B can be distinguished by A.

Theorem 3 lays out a critical fact on the relations between GT test and structural encodings: *if $S' \succeq S$, then compared with S-GT, S'-GT is more powerful in graph isomorphism testing and will always converge faster when testing graphs.* Through Theorem 3 , we can distinguish the expressive power of various structural encodings by comparing them with baseline encodings defined in Section 4.1. Given existing structural encodings, Theorem 3 shows that more powerful encodings can be developed by adding extra non-trivial structural information. We will elaborate on the ideas above in the following text.

4.3 WL AS A SPECIAL CASE OF GT TEST

The first application of our theoretical results is to answer the question: *How to design graph Transformers that are more powerful than the WL test?* Since the expressivity of graph Transformers depends on the corresponding GT test, we first characterize WL test as a special case of GT test:

Theorem 4. *Two non-isomorphic graphs can be distinguished by WL if and only if they are distinguishable by Neighbor-GT.*

Theorem 4 proves that though *Neighbor*-GT hashes the whole set of node labels, its expressivity is still exactly the same as WL test. Therefore, from a theoretical perspective, graph Transformer models with *Neighbor* encoding have the same expressive power as WL-GNNs, though they feature the multi-head attention mechanism and global receptive field for every node. Combined with Theorem 3, the answer to the question above becomes simple: *To design a graph Transformer that is more powerful than the WL test, we only need to equip it with structural encoding more expressive than Neighbor.*

Furthermore, considering many GNNs utilize absolute structural encodings to enhance their expressive power (e.g., Bouritsas et al. (2022)), we wonder how to compare their expressiveness against Transformers. For any absolute structural encoding f_A, we can easily infer from Theorem 4 that f_A-WL (WL with additional node features generated by f_A) is equivalent to $(f_A, \mathit{Neighbor}_R)$-GT on expressive power. Therefore, to develop graph Transformers with expressivity beyond WL-GNNs, it is necessary to design relative structural encodings that are more powerful than $\mathit{Neighbor}_R$.

5 SHORTEST-PATH-BASED RELATIVE STRUCTURAL ENCODINGS

This section presents an example of utilizing our theory and designing powerful relative structural encodings for graph Transformers. We start from encodings based on the shortest path between two nodes, like *SPD* used in Graphormer (Ying et al., 2021).

5.1 EXPRESSIVITY OF *SPD* ENCODING

Considering that two nodes are adjacent when SPD between them is 1, we can easily conclude that $\mathit{SPD}_R \succeq \mathit{Neighbor}_R$. Therefore, it can be inferred from Theorem 3 that *SPD*-GT is more powerful than WL. Besides, we can find many pairs of non-isomorphic graphs indistinguishable by WL but not for *SPD*-GT. We have

Theorem 5. *(1) SPD-GT is **strictly** more expressive than WL in testing non-isomorphic graphs[2]; (2) For G_1 and G_2 that WL distinguishes as non-isomorphic after t iterations, SPD-GT can distinguish G_1 and G_2 as non-isomorphic within t iterations.*

Theorem 5 formally proves that *SPD*-GT is strictly more powerful and converges faster than WL in graph isomorphism testing. In addition to Theorem 5, we want to find out how the global structural information leveraged by shortest path encodings affects the discriminative power of GT test. We introduce the concept of *receptive field* of structural encodings, that when S has k-hop receptive field, any structural information encoded by S only depends on the k-hop neighborhood of the central node. For example, *Neighbor* has 1-hop receptive field because only neighborhood connections are considered by *Neighbor* encoding. However, the receptive field of *SPD* is not restricted to k-hop for any k, since we can construct graphs with SPD between two nodes arbitrarily large. We show this global-aware receptive field brings distinctive power to *SPD* that differs from any encodings with local receptive field, in following Proposition 1:

[2]For two isomorphic testing algorithms A and B, we say A is strictly more expressive than B if A is more expressive than B in testing non-isomorphic graphs, and there exist non-isomorphic graphs G_1 and G_2 such that A can distinguish G_1 and G_2 but not for B.

Proposition 1. *For any k and any structural encoding scheme S with k-hop receptive field, there exists a pair of graphs that SPD-GT can distinguish, but S-GT can not.*

Though *SPD* has its unique expressive power and is more powerful than WL, many low-order non-isomorphic graphs remain to be indistinguishable by *SPD*-GT (see Proof for Theorem 6), which leads us to find encodings that are more powerful than *SPD*. Following Theorem 3, building structural encoding S that satisfies $S \succeq$ *SPD* can be done by adding meaningful information to *SPD*, which illustrates the motivation for *SPIS* we will next introduce.

5.2 *SPIS* Relative Structural Encoding

From the perspective of graph theory, for two connected nodes v, u in the graph, there can be multiple shortest paths connecting v and u, and these shortest paths may be linked or have overlapping nodes. Since *SPD* only encodes the length of shortest paths, one intuitive idea is to enhance it with features characterizing the rich structural interactions between different shortest paths. Inspired by concepts like betweenness centrality in network analysis (Freeman, 1977), we propose the concept of shortest path induced subgraph (SPIS) to characterize the structural relations between nodes on shortest paths:

Definition 5.1 (Shortest Path Induced Subgraph)**.** *For $G = (V, E)$ and $v, u \in V$, $SPIS(v, u) = (V_{SPIS(v,u)}, E_{SPIS(v,u)})$, the **shortest path induced subgraph** between v and u is an induced subgraph of G, where*

$$V_{SPIS(v,u)} = \{s : s \in V \text{ and } SPD_R(v, s) + SPD_R(s, u) = SPD_R(v, u)\}. \tag{9}$$

$\text{SPIS}(v, u)$ is an induced subgraph of G that contains all nodes on shortest paths between v and u. To encode knowledge in SPIS as numerical vectors, we propose the relative encoding method $SPIS_R$ by enhancing SPD_R with the total numbers of nodes and edges of SPIS between nodes, as

$$SPIS_R(v, u, G) = (SPD_R(v, u, G), |V_{SPIS(v,u)}|, |E_{SPIS(v,u)}|), \tag{10}$$

and we define the structural encoding scheme $SPIS = (id_A, SPIS_R)$.

5.3 Analysis on *SPIS* Encoding

In the following, we will analyze the proposed *SPIS* encoding and characterize its mathematical properties, comparing it with *SPD* and WL. To start with, as *SPIS* is constructed by adding information to *SPD*, we have *SPIS* $\succeq$ *SPD* and it is be more powerful than *SPD*-GT according to Theorem 3.

Theorem 6. *(1) SPIS-GT is **strictly** more expressive than SPD-GT in testing non-isomorphic graphs.*

(2) For G_1 and G_2 that SPD-GT distinguishes as non-isomorphic after t iterations, SPIS-GT can distinguish G_1 and G_2 as non-isomorphic within t iterations.

Next, we show that *SPIS*-GT exhibits far superior performance to WL and *SPD*-GT on important graph structures. The computational complexity of *SPIS* is discussed in Appendix C.

***SPIS*-GT Distinguishes All Low-order Graphs ($n \leq 8$).** On low-order graphs, our synthetic experiments in Table 1 confirm that *SPIS*-GT distinguishes *all* non-isomorphic graphs with order equal to or less than 8, which is much more powerful than WL with 332 indistinguishable pairs and *SPD*-GT with 200 indistinguishable pairs. This strong discriminative power on low-order graphs shows that *SPIS* can accurately distinguish local structures in real-world graphs.

***SPIS*-GT Well Distinguishes Strongly Regular Graphs.** A regular graph is a graph parameterized by two parameters n, k which has n nodes and each node has the k neighbors, denoted as $\text{RG}(n, k)$. And a strongly regular graph parameterized by four parameters (n, k, λ, μ) is a regular graph $\text{RG}(n, k)$ where every adjacent pair of nodes has the same number λ of neighbors in common, and every non-adjacent pair of nodes has the same number μ of neighbors in common, denoted as $\text{SRG}(n, k, \lambda, \mu)$.

Due to their highly symmetric structure, regular graphs are known to be failure cases for graph isomorphism test algorithms. For example, WL can not discriminate any regular graphs of the same parameters, making any pair of strongly regular graphs with the same n and k indistinguishable to it, even λ and μ could be different. Yet Proposition 2 guarantees that *SPIS*-GT can distinguish any pair of strongly regular graphs of different parameters:

Proposition 2. *SPIS-GT can distinguish any pair of strongly regular graphs of different parameters.*

	Low-Order Graphs (Parameter: n)				Strongly Regular Graphs (Parameter: (n, k, λ, μ))					
Parameter	5	6	7	8	$(25, 12, 5, 6)$	$(26, 10, 3, 4)$	$(29, 14, 6, 7)$	$(36, 14, 4, 6)$	$(40, 12, 2, 4)$	$(45, 12, 3, 3)$
# Graphs	21	112	853	11117	15	10	41	180	28	78
# Graph Pairs	210	6216	363378	61788286	105	45	820	16110	378	3003
Method	# Indistinguishable Graph Pairs									
WL	0	3	17	312	105	45	820	16110	378	3003
SPD-GT	0	2	12	186	105	45	820	16110	378	3003
SPIS-GT	0	**0**	**0**	**0**	**0**	**0**	**0**	**15**	**3**	**0**

Table 1: Results of synthetic graph isomorphism tests.

It is worth mentioning that, for strongly regular graphs with the same parameters, ***SPIS*** also exhibits outstanding discriminative power, with the number of total failures being far less than WL and *SPD*-GT (See Section 6.1 and Table 1).

***SPIS*-GT Distinguishes 3-WL Failure Cases.** When compared with k-order WL tests ($k \geq 3$, GT test costs only $O(n^2)$ time complexity at each iteration, and the flexible choice of structural encoding method allows it to show a wide range of expressive capabilities. Here, we show that *SPIS*-GT is able to distinguish a pair of graphs that 3-WL can not distinguish:

Proposition 3. *There exists a pair of graphs that SPIS-GT can distinguish, but 3-WL can not.*

6 Experiments

In this section, we first perform synthetic isomorphism tests on low order graphs and strongly regular graphs to evaluate the expressive power of proposed ***SPIS*** encoding against several previous benchmark methods. Then we show that by replacing ***SPD*** encoding with the provably stronger ***SPIS***, the performance of the well-tested Graphormer model on a wide range of real-world datasets can be significantly improved.

6.1 Synthetic Isomorphism Tests

Settings. To evaluate the structural expressive power of WL test and GT test with structural encodings described above, we first perform synthetic isomorphism tests on a collection of connected low-order graphs up to 8 nodes and strongly regular graphs up to 45 nodes[3]. We run the algorithms above and check how they can disambiguate non-isomorphic low order graphs with the same number of nodes and strongly regular graphs with the same parameters. The results are shown in Table 1.

Results. For low order graphs, results in Table 1 show that *SPD*-GT can distinguish more non-isomorphic graphs than WL, but neither can match the effectiveness of *SPIS*-GT which disambiguates any low-order graphs up to 8 nodes. As for the highly symmetric strongly regular graphs, both WL and *SPD*-GT cannot discriminate any strongly regular graphs with the same parameters, yet *SPIS*-GT only has few indistinguishable pairs. Compared with WL and *SPD*-GT, *SPIS*-GT has outstanding structural expressive power. Since many real-world graphs (like molecular graphs) consist of small motifs with highly symmetrical structures, it is reasonable to expect that graph Transformers with ***SPIS*** can accurately capture significant graph structures and exhibit strong discriminative power.

6.2 Graph Representation Learning

Datasets. To test the real-world performance of graph Transformers with proposed structural encodings, we select 8 popular graph representation learning benchmarks: 4 property regression datasets (ZINC(subset) (Irwin & Shoichet, 2005; Dwivedi et al., 2020), QM9, QM8, ESOL (Wu et al., 2018)) and 4 classification datasets (PTC-MR, MUTAG, COX2, PROTEINS (Morris et al., 2020a)). Appendix E.1 provides detailed descriptions for the datasets.

Settings and Baselines. To investigate how the expressive power of structural encodings affects the benchmark performance of real graph Transformers, we choose the Graphormer (Ying et al., 2021) as the backbone model for testing structural encoding since (i) Graphormer proposes SPD, which we have characterized and has expressivity stronger than WL; (ii) the way Graphormer introduces relative structural encodings can correspond to our Theorem 2 which analyzes a simple Transformer network incorporating relative encodings via attention biases; (iii) Graphormer's code is well-constructed and

[3]We use the database in `http://www.maths.gla.ac.uk/~es/srgraphs.php` to collect strongly regular graphs with the same set of parameters.

Task	Regression				Classification			
Dataset	ZINC	QM9	QM8	ESOL	PTC-MR	MUTAG	COX2	PROTEINS
Metric	MAE↓	Multi-MAE↓		RMSE↓	Accuracy↑			
Method	Results							
GCN	0.469±0.002	1.006±0.020	0.0279±0.0001	0.564±0.015	67.97±6.49	85.76±8.75	80.42±5.23	76.00±3.20
GAT	0.463±0.002	1.112±0.018	0.0317±0.0001	0.552±0.007	67.21±2.50	84.59±6.30	79.36±7.23	71.15±7.12
GIN	0.408±0.008	1.225±0.055	0.0276±0.0001	0.626±0.017	68.27±5.11	89.40±5.40	82.57±4.55	75.90±2.80
GraphSAGE	0.410±0.005	0.855±0.002	0.0275±0.0001	0.601±0.008	60.53±5.24	85.10±7.60	78.07±7.07	75.90±3.20
GSN	0.140±0.006	-	-	-	67.40±5.70	92.20±7.50	-	74.60±5.00
PNA	0.320±0.032	-	-	-	-	-	-	-
1-2-3-GNN	-	-	-	-	60.90	86.10	-	75.50
MoleculeNet	-	2.350	0.0150	0.580	-	-	-	-
WL	-	-	-	-	59.90±4.30	90.40±5.70	-	75.00±3.10
RetGK	-	-	-	-	62.50±1.60	90.30±1.10	80.10±0.90	76.20±0.50
P-WL	-	-	-	-	64.02±0.82	90.51±1.34	-	75.31±0.73
FGW	-	-	-	-	65.31±7.90	88.42±5.67	77.23±4.86	74.55±2.74
GT	0.226±0.01	-	-	-	-	-	-	-
SAN	0.139±0.01	-	-	-	-	-	-	-
SAT	0.135	-	-	-	-	-	-	-
Graphormer-*id*	0.668±0.003	3.176±0.005	0.0144±0.0003	0.612±0.002	66.39±5.18	85.49±8.51	77.60±7.69	77.19±4.07
Graphormer-*Neighbor*	0.531±0.004	1.799±0.002	0.0141±0.0002	0.639±0.034	68.13±6.82	90.35±7.01	78.06±7.43	78.12±3.62
Graphormer-*SPD*	0.122±0.001	0.607±0.002	0.0079±0.0001	0.492±0.004	68.43±5.82	91.39±7.35	82.12±3.40	78.59±4.35
Graphormer-*SPIS*	**0.115±0.001**	**0.595±0.001**	**0.0073±0.0001**	**0.484±0.005**	**69.28±5.34**	**92.48±5.87**	**83.22±2.25**	**79.41±1.46**

Table 2: Results of graph representation learning benchmarks. All results except for GCN, GAT, GIN, GraphSAGE, SAT and Graphormer variants are cited from their original papers. ↓ for lower is better, and ↑ for higher is better. Appendix E.3 reports performances on QM9 by seperate tasks.

publicly available. The original Graphormer utilizes a SPD_R relative structural encoding (discussed in Appendix D), so we name it as Graphormer-*SPD*. We build a new Graphormer-*SPIS* model by replacing the SPD_R encoding with $SPIS_R$ encoding as an improved version of Graphormer while keeping other network components unchanged. Similarly, we also use Graphormer-*id* and Graphormer-*Neighbor* as less expressive Graphormer variants.

In addition, we compare the above Graphormer variants against (i) GNNs including GCN (Kipf & Welling, 2016), GIN (Xu et al., 2018), GAT (Veličković et al., 2017), GraphSAGE (Hamilton et al., 2018), GSN (Bouritsas et al., 2022), PNA (Corso et al., 2020) and 1-2-3-GNN (Morris et al., 2019); (ii) best performances collected by MoleculeNet paper (Wu et al., 2018); (iii) graph kernel based methods including WL subtree kernel (Shervashidze et al., 2011), RetGK (Zhang et al., 2018), P-WL (Rieck et al., 2019) and FGW (Titouan et al., 2019); (iv) graph Transformers including GT (Dwivedi & Bresson, 2020), SAN (Kreuzer et al., 2021) and SAT (Chen et al., 2022). One can find the detailed descriptions of Graphormer variants, baselines, and training settings in Appendix E.2.

Results. Table 2 presents the results of graph representation learning benchmarks. It can be observed that the performances of Graphormer variants mostly align with their relative ranking of expressive power ($SPIS \succeq SPD \succeq Neighbor \succeq id$), and replacing the *SPD* encoding in Graphormer with the proposed stronger *SPIS* encoding results in a consistent performance improvement, demonstrating that real-world performance of graph Transformers can benefit from theoretically expressive structural encoding designs. Equipped with the provably powerful *SPIS* encoding, Graphormer-*SPIS* achieves state-of-the-art performance and outperforms existing graph Transformers and GNNs, which echoes our theoretical results on the strong expressive power of *SPIS*. Meanwhile, when employing the less expressive *Neighbor* and *id* encoding, the Transformer network loses the ability to accurately distinguish graph structures, leading to a significant performance drop.

7 Conclusion

In this paper, we introduce GT test as a novel unified framework for analyzing the expressive power of graph Transformers. In this framework, we theoretically characterize how to improve the expressivity of graph Transformers with respect to WL test and GNNs, and propose a provably powerful structural encoding method *SPIS*. Experiments have verified that the performances of benchmark graph Transformers can benefit from this theory-oriented extension. We also discuss our work's limitations and potential social impact in Appendix F.

REPRODUCIBILITY STATEMENT

Proofs for all theoretical results in this paper are provided in the Appendix. Aside from baseline results cited from existing papers, code and instructions for reproducing the experiments of proposed method is provided in the supplementary material.

REFERENCES

Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On weisfeiler-leman invariance: Subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 113:42–59, 2020.

Waiss Azizian and Marc Lelarge. Expressive power of invariant and equivariant graph neural networks. *arXiv preprint arXiv:2006.15646*, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos P Zafeiriou, and Michael Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

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

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Robert W Floyd. On ambiguity in phrase structure languages. *Communications of the ACM*, 5(10): 526, 1962.

Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pp. 35–41, 1977.

Martin Grohe. *Descriptive complexity, canonisation, and definable graph structure theory*, volume 47. Cambridge University Press, 2017.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

John J Irwin and Brian K Shoichet. Zinc- a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1):177–182, 2005.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Devin Kreuzer, Dominique Beaini, William L Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *arXiv preprint arXiv:2106.03893*, 2021.

Xin Liu, Haojie Pan, Mutian He, Yangqiu Song, Xin Jiang, and Lifeng Shang. Neural subgraph isomorphism counting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1959–1969, 2020.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Federico Monti, Karl Otness, and Michael M Bronstein. Motifnet: a motif-based graph convolutional network for directed graphs. In *2018 IEEE Data Science Workshop (DSW)*, pp. 225–228. IEEE, 2018.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020a.

Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. *Advances in Neural Information Processing Systems*, 33: 21824–21840, 2020b.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Raghunathan Ramakrishnan, Mia Hartmann, Enrico Tapavicza, and O Anatole Von Lilienfeld. Electronic spectra from tddft and machine learning in chemical space. *The Journal of chemical physics*, 143(8):084111, 2015.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer. *arXiv:2205.12454*, 2022.

Bastian Rieck, Christian Bock, and Karsten Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In *International Conference on Machine Learning*, pp. 5448–5458. PMLR, 2019.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *arXiv preprint arXiv:2007.02835*, 2020.

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 333–341. SIAM, 2021.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Vayer Titouan, Nicolas Courty, Romain Tavenard, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pp. 6275–6284. PMLR, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Clement Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. *Advances in Neural Information Processing Systems*, 33:14143–14155, 2020.

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

Asiri Wijesinghe and Qing Wang. A new perspective on" how graph neural networks go beyond weisfeiler-lehman?". In *International Conference on Learning Representations*, 2021.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? *arXiv preprint arXiv:2106.05234*, 2021.

Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. *arXiv preprint arXiv:2101.10320*, 2021.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Zhen Zhang, Mianzhi Wang, Yijian Xiang, Yan Huang, and Arye Nehorai. Retgk: Graph kernels based on return probabilities of random walks. *Advances in Neural Information Processing Systems*, 31, 2018.

Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094*, 2021.

A PROOFS

A.1 THEOREM 1

We first restate Theorem 1 in a more generalized version which can be applied to both cases when the graph embedding is computed by a global readout function or virtual node trick:

Theorem 1. *For any structural encoding scheme $S = (f_A, f_R)$ and labeled graph $G = (V, E)$ with label map $h_0 : V \to \mathcal{X}$, if a graph neural model $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ satisfies the following conditions:*

1. *$\mathcal{A}$ computes the initial node embeddings with*

$$l_0(v) = \phi(h_0(v), f_A(v, G)), \tag{11}$$

2. *$\mathcal{A}$ aggregates and updates node embeddings iteratively with*

$$l_t(v) = \sigma(\{\!\{(l_{t-1}(u), f_R(v, u, G)) : u \in V\}\!\}), \tag{12}$$

   where ϕ and σ above are model-specific functions,

3. *The final graph embedding is computed by a global readout on the multiset of node features $\{\!\{l_t(v) : v \in V\}\!\}$, or represented by the embedding of node s such that for any $u, v \in V$, $f_R(s, v, G) = f_R(s, u, G) = f_R(v, s, G) = f_R(u, s, G)$.*

then for any labeled graphs G_1 and G_2, if $\mathcal{A}$ maps them to different embeddings, S-GT also decides G_1 and G_2 are not isomorphic.

Proof. We first show that for any node v, u at iteration t, if S-GT generates $g_t(v) = g_t(u)$, then $\mathcal{A}$ also generates the same embeddings for v and u as $l_t(v) = l_t(u)$. For $t = 0$ this proposition holds because if $g_0(v) = g_0(u)$ then v and u must have the same input label and absolute structural encoding, which leads to $l_0(v) = l_0(u)$. Suppose this proposition holds for iteration $0, 1, \ldots, t$ and $g_{t+1}(v) = g_{t+1}(u)$. From the injectiveness of function Φ, we have

$$\{\!\{(g_t(r), f_R(v, r, G)) : r \in V_v\}\!\} = \{\!\{(g_t(r), f_R(u, r, G)) : r \in V_u\}\!\}, \tag{13}$$

where V_v is the node set of graph that v belongs to, which is the same for V_u. If two finite multisets are identical, then the elements in the two multisets can be matched in pairs. Therefore, according to our assumption at iteration t such that $g_t(v) = g_t(u) \implies l_t(v) = l_t(u)$, we have

$$\{\!\{(l_t(r), f_R(v, r, G)) : r \in V\}\!\} = \{\!\{(l_t(r), f_R(u, r, G)) : r \in V\}\!\}. \tag{14}$$

Considering $\mathcal{A}$ updates node labels by $l_{t+1}(v) = \sigma(\{\!\{(l_t(r), f_R(v, r, G)) : r \in V\}\!\})$, $l_{t+1}(v) = l_{t+1}(u)$ holds. This proves the proposition above by induction. Now that for any iteration t we have $g_t(v) = g_t(u) \implies l_t(v) = l_t(u)$, indicating that a mapping ψ_t exists such that for any node v, $l_t(v) = \psi_t(g_t(v))$.

Now consider two graphs G_1 and G_2 where $\mathcal{A}$ maps them to different embeddings after t iterations. If $\mathcal{A}$ computes the graph embedding by a readout function on the multiset of node features, then $\{\!\{l_t(r) : r \in V\}\!\}$ must be different for two graphs. Since $\{\!\{l_t(r) : r \in V\}\!\} = \{\!\{\psi_i(g_t(r)) : r \in V\}\!\}$, $\{\!\{g_t(r) : r \in V\}\!\}$ must also be different for two graphs, which shows that S-GT decides G_1 and G_2 are not isomorphic. Meanwhile, if the graph embedding is represented by embedding of node s such that for any $u, v \in V$, $f_R(s, v, G) = f_R(s, u, G) = f_R(v, s, G) = f_R(u, s, G)$, then $l_t(s)$ is different for two graphs. Since $l_t(s)$ is generated by $l_t(s) = \sigma(\{\!\{(l_{t-1}(r), f_R(s, r, G)) : r \in V\}\!\})$ and $f_R(s, r, G)$ is the same for every $r \in V$, $\{\!\{l_{t-1}(r) : r \in V\}\!\}$ must be different for two graphs, which goes back to the situation we have discussed above. Therefore, the proof is completed. □

A.2 THEOREM 2

Our proof for Theorem 2 is largely based on the proof for the universal approximation theorem of the Transformer architecture, so it is strongly recommended to go through the proof in Yun et al. (2019) before reading our proof in the next section.

A.2.1 BIAS-GT MODEL

To present a simple and flexible example on building theoretically powerful graph Transformers, we propose bias-GT, a graph Transformer model that works under any structural encoding schemes with minimal modifications to the original Transformer architecture. More concretely, for $S = (f_A, f_R)$ and input graph G, the input embedding of node v is computed by

$$l_0(v) = \text{Linear}(\text{Concat}(h_0(v), f_A(v, G)), \tag{15}$$

where $\text{Linear}(\cdot)$ is a linear layer, $\text{Concat}(\cdot)$ refers to the concatenation operation. At every Transformer layer, the relative structural encodings are introduced as transformed attention biases. For every node pair (u, v), the final attention weight a_{uv} from node u to v is computed by

$$a_{uv} = \bar{a}_{uv} + \text{Embedding}(f_R(u, v, G)), \tag{16}$$

where $\bar{a}_{uv}$ is the original attention weight computed by scaled-dot self-attention, and $\text{Embedding}(\cdot)$ transforms relative embeddings in $\mathcal{C}$ to $\mathbb{R}$ using via embedding lookup or linear layers. All remaining network components stay the same with the original Transformer architecture. This bias-GT model offers a straightforward strategy for injecting strutural information to the Transformer and can be viewed as a simplified version of some exisiting models (Ying et al., 2021; Zhao et al., 2021). We will use S-bias-GT to denote bias-GT network with structural encoding scheme S. The proposition below shows that S-GT test limits the expressive power of S-bias-GT:

Proposition 4. *For any regular structural encoding scheme $S = (f_A, f_R)$ and two graphs G_1, G_2, if S-bias-GT maps them to different embeddings, S-GT also decides G_1 and G_2 are not isomorphic.*

Proof. We only need to check the conditions in Theorem 1. For the first condition, S-bias-GT computes the initial node embeddings with

$$l_0(v) = \phi(h_0(v), f_A(v, G)) = \text{Linear}(\text{Concat}(\bar{h}_0(v), f_A(v, G)), \tag{17}$$

and for the second condition, since S is regular, the relative structural encoding functions satisfy $f_R(v, v, G) \neq f_R(v, u, G)$ for $v, u \in V$, then a function operated on $\{\!\{(l_{t-1}(u), f_R(v, u, G)) : u \in V\}\!\}$ can be viewed as a function operated on $(h_v, \{\!\{(l_{t-1}(u), f_R(v, u, G)) : u \in V\}\!\})$ because $f_R(v, v, G)$ is different from all other relative encodings. S-bias-GT updatesthe node embeddings with

$$\begin{aligned} l_t(v) =& \sigma(h_v, \{\!\{(l_{t-1}(u), f_R(v, u, G)) : u \in V\}\!\}) && (18)\\ =& \text{FFN}(\text{Concat}_{i=1,\ldots,h}(\sum_{u \in V} w_{vu}^i l_{t-1}(v) W_Q^i) W_O), && (19)\end{aligned}$$

$$\text{where } w_{vu}^i = \frac{\exp(\bar{\alpha}_{vu}^i)}{\sum_{r \in V} \exp(\bar{\alpha}_{vr}^i)} \tag{20}$$

$$\text{and } \bar{\alpha}_{vu}^i = \frac{(l_{t-1}(v) W_Q^i)(l_{t-1}(u) W_K^i)^\top}{\sqrt{d}} + \text{Embedding}_i(f_R(v, u, G)). \tag{21}$$

W_Q^i, W_K^i, W_V^i, W_O above are projection matrices, FFN is the feed-forward layer, and layer normalization and residual connections are omitted for clarity. The function σ is basically the computation steps of the Transformer with $f_R(v, u, G)$ injected as attention bias. Since the graph embedding can be computed by a global readout function, according to Theorem 1, the proof is completed. □

A.2.2 EXPLAINATIONS ON THEOREM 2

When the input graph order n is fixed, let the input be $G = (V, E)$ with label map h_0 and $V = \{v_1, \ldots, v_n\}$. To properly define this approximation process, for some structural encoding scheme S, the input for both GT test and bias-GT network g is viewed as the feature matrix $\boldsymbol{X}_0 = [h_0(v_1), \ldots, h_0(v_n)] \in \mathbb{R}^{d \times n}$ and the adjacency matrix $\boldsymbol{A}$ with permutation invariance. We define the GT test function f_t by stacking all labels generated by t-iteration GT test in $f_t(\boldsymbol{X}_0, \boldsymbol{A}) = [g_t(v_1), \ldots, g_t(v_n)]$, and the output of g is similarly defined by stacking all feature vectors generated by the network. We define the d_p distance between f_t and g as $\mathsf{d}_p(f_t, g) = \max_{\boldsymbol{A}} \mathsf{d}_p(f_t(\cdot, \boldsymbol{A}), g(\cdot, \boldsymbol{A}))$, where $\mathsf{d}_p(f_t(\cdot, \boldsymbol{A}), g(\cdot, \boldsymbol{A}))$ is the ℓ^p distance on $\mathbb{R}^{d \times n}$ between f_t and g when $\boldsymbol{A}$ is fixed. Following (Yun et al., 2019), d_p also stands for ℓ^p distance between functions in the remaining context.

$\Phi(\{\!\{(g_{t-1}(v_i), f_R(v_j, v_i, G)) : v_i \in V)\}\!\})$ can be viewed as a permutation (of node order) invariant function $\Phi(\boldsymbol{X}_{t-1}, \boldsymbol{W}_j)$, where $\boldsymbol{X}_{t-1} = [g_{t-1}(v_1), \ldots, g_{t-1}(v_n)] \in \mathbb{R}^{d\times n}$ is the matrix of node labels and $\boldsymbol{W}_j = [f_R(v_j, v_1, G)), \ldots, f_R(v_j, v_n, G))] \in \mathcal{C}^n$. The assumption that Φ can be extended to a continuous function with respect to node labels means that, for any fixed $\boldsymbol{W}_j$, $\Phi(\boldsymbol{X}_{t-1}, \boldsymbol{W}_j)$ is a continuous function with respect to any entry-wise ℓ^p norm of $\boldsymbol{X}_{t-1}$ with compact support in $\mathbb{R}^{d\times n}$ (since $\mathcal{X}$ is compact).

A.2.3 Proof for Theorem 2

Proof. Since the first iteration of GT test can be arbitrarily approximated by performing a linear layer on embeddings generated by concatnating the initial embeddings and absolute positional encodings, according to the universal approximation theorem (Hornik et al., 1989) and Lipschitz continuity of feed-forward layers, the key technical challenge in proving Theorem 2 is showing that each iteration i in $1, \ldots, t$ of GT test can be approximated arbitrarily well using the bias-GT network. Let f stands for one iteration of S-GT test, with input and output defined according to f_t. We denote $\boldsymbol{\alpha}_{i,j} = f_R(v_i, v_j, G)$ for simplicity, and let $\boldsymbol{x}_i$ be the input labels of v_i for f. Then f can be viewed as:

$$f(\boldsymbol{X}, \boldsymbol{A}) = [\Phi(\{\!\{(\boldsymbol{x}_i, \boldsymbol{\alpha}_{1,i})\}\!\}_{i=1,\ldots,n}), \ldots, \Phi(\{\!\{(\boldsymbol{x}_i, \boldsymbol{\alpha}_{n,i})\}\!\}_{i=1,\ldots,n})]. \tag{22}$$

That is, if our Transformer network is capable of approximating the multiset function Φ that takes the feature matrix and structural encodings as input, then it can approximate f at any precision because the output of f contains n entries computed individually by Φ. As we have mentioned, Φ can be rewritten to the following equivalent form:

$$\Phi(\{\!\{(\boldsymbol{x}_i, \boldsymbol{\alpha}_{j,i})\}\!\}_{i=1,\ldots,n}) = \Phi(\boldsymbol{X}, \boldsymbol{W}_j), \text{ where } \boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \text{ and } \boldsymbol{W}_j = [\boldsymbol{\alpha}_{j,1}, \ldots, \boldsymbol{\alpha}_{j,n}]. \tag{23}$$

In this form, Φ is permutation equivariant such that for any permutation matrix $\boldsymbol{P}$, $\Phi(\boldsymbol{X}\boldsymbol{P}, \boldsymbol{W}_j\boldsymbol{P}) = \Phi(\boldsymbol{X}, \boldsymbol{W}_j)$.

The major problem is that the bias-GT network g only take $\boldsymbol{X}$ as feature input while incorporating structural encodings $\boldsymbol{W}_j$ in attention layers as biases. According to our assumptions, we can assume without generality that $\mathcal{X} \subset (0, 1)^d$ and the compact support of extented function Φ with respect to $\boldsymbol{X}$ is contained within $[0, 1]^{d\times n}$. We follow the proof structure outlined in Yun et al. (2019).

Step 1: Approximate f by $\bar{f}$, a piece-wise constant function with respect to $\boldsymbol{X}$. According to previous assumptions and statements in Yun et al. (2019), for any fixed $\boldsymbol{W}_j$, Φ is a uniform continuous function (because Φ has compact support) with respect to the argument $\boldsymbol{X}$. Suppose for $\boldsymbol{W}_j$, there exists $\delta_{\boldsymbol{W}_j}$ such that for any $\boldsymbol{X}, \boldsymbol{Y}$,$\|\boldsymbol{X} - \boldsymbol{Y}\|_\infty < \delta_{\boldsymbol{W}_j}$ we have $\|\Phi(\boldsymbol{X}, \boldsymbol{W}_j) - \Phi(\boldsymbol{Y}, \boldsymbol{W}_j)\|_p < \frac{\epsilon}{3}$. Since the possible graph structures of order n is finite, $\mathcal{C}$ is a finite set and the possible choices of $\boldsymbol{W}_j$ is also finite. Therefore, we can pick $\delta = \min_{\boldsymbol{W}_j}\{\delta_{\boldsymbol{W}_j}\}$, then for any $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{W}_j$, if $\|\boldsymbol{X} - \boldsymbol{Y}\|_\infty < \delta$ we have $\|\Phi(\boldsymbol{X}, \boldsymbol{W}_j) - \Phi(\boldsymbol{Y}, \boldsymbol{W}_j)\|_p < \frac{\epsilon}{3}$. Accordingly, we can define a piece-wise constant function $\bar{\Phi}$ to approximate Φ as

$$\bar{\Phi}(\boldsymbol{X}, \boldsymbol{W}_j) = \sum_{\boldsymbol{L}\in\mathbb{G}_\delta} \Phi(\boldsymbol{C}_{\boldsymbol{L}}, \boldsymbol{W}_j)\mathbb{1}\{\boldsymbol{X} \in \mathbb{S}_{\boldsymbol{L}}\}, \tag{24}$$

where $\mathbb{S}_{\boldsymbol{L}}$ is a cube of width δ with $\boldsymbol{L}$ being one of its vertices, $\boldsymbol{C}_{\boldsymbol{L}} \in \mathbb{S}_{\boldsymbol{L}}$ is the center point of $\mathbb{S}_{\boldsymbol{L}}$ (Please refer to Appendix B.1 of (Yun et al., 2019) for a detailed explanation). By the uniform continuity of Φ, we can prove $\|\Phi(\boldsymbol{X}, \boldsymbol{W}_j) - \bar{\Phi}(\boldsymbol{X}, \boldsymbol{W}_j)\|_p < \frac{\epsilon}{3}$ for any $\boldsymbol{X}, \boldsymbol{W}_j$. Also, it is trivial to verify that $\bar{\Phi}$ is permutation equivariant. By defining $\bar{f}$ by replacing function Φ with $\bar{\Phi}$ in Equation 22, we have $\mathsf{d}_p(f, \bar{f}) \leq \frac{\epsilon}{3}$.

Step 2: Approximate $\bar{f}$ with modified bias-GT network. In this step we aim to approximate $\bar{f}$ using a modified bias-GT network, where the softmax operator $\sigma[\cdot]$ and $\mathrm{ReLU}(\cdot)$ are replaced by the γ-hardmax operator $\sigma_{\mathrm{H},\gamma}[\cdot]$ and an activation finction ϕ that is a piece-wise linear function with at most three pieces in which at least one piece is constant. Note that the γ-hardmax operator is defined by adding $\gamma > 0$ to non-zero elements of σ_{H}.

Proposition 5. *$\bar{\Phi}$ can be approximated by a modified bias-GT network $\bar{g}$ such that $\mathsf{d}_p(\bar{f}, \bar{g}) \leq \frac{\epsilon}{3}$.*

Step 3: Approximate modified bias-GT network with (original) bias-GT network. Finally, we will show that the modified bias-GT $\bar{g}$ can be approximated by the original bias-GT architecture.

Proposition 6. *$\bar{g}$ can be approximated by a bias-GT network g such that $\mathsf{d}_p(g, \bar{g}) \leq \frac{\epsilon}{3}$.*

Following (Yun et al., 2019), along with three steps above, we prove that a single S-GT iteration f can be arbitrarily approximated with a bias-GT network g. By stacking such bias-GT networks, we show that S-GT with any number of iterations can be approximated by S-bias-GT at any precision. We next provide proofs for the two propositions. □

A.2.4 Proof for Proposition 6

Proof. We only need to notice that for any $\boldsymbol{A}$, $\sigma_{\mathrm{H},\gamma}(\boldsymbol{A}) \to \sigma_{\mathrm{H}}(\boldsymbol{A})$ as $\gamma \to 0$. Then together with Appendix B.2 of Yun et al. (2019), we can finish the proof. □

A.2.5 Proof for Proposition 7

Proof. We will prove this statement in five major steps:

1. Given input $\boldsymbol{X}$, a group of feed-forward layers in the modified Transformer network can quantize $\boldsymbol{X}$ to an element $\boldsymbol{L}$ on the grid $\mathbb{G}_\delta := \{0, \delta, \ldots, 1-\delta\}^{d\times n}$.
2. A group of additional feed-forward layers then scales $\boldsymbol{L}$ to a different level, where for every $l_j := \boldsymbol{u}^\top \boldsymbol{L}_{:,j}$, $l_j \in \{1, \delta^{-1}, \delta^{-2}, \ldots, \delta^{-\delta^{-d}+1}\}$ holds. ($\boldsymbol{u} = (1, \delta^{-1}, \delta^{-2}, \ldots, \delta^{-d+1})$.)
3. A group of biased self-attention layers perform global shift on $\boldsymbol{L}$, such that for any i and j, the shifted l_i and l_j are different if and only if their corresponding multisets of label-RSE tuples ($\{\!\{(\boldsymbol{x}_k, \boldsymbol{\alpha}_{i,k}) : k = 1, \ldots, n\}\!\}$ for l_i) are different.
4. Next, a group of self-attention layers map the shifted $\boldsymbol{L}$ to the desirable **contextual mappings** $q(\boldsymbol{L})$. (defined in Yun et al. (2019))
5. Finally, a group of feed-forward layers can map elements of the contextual embeddings $q(\boldsymbol{L})$ to the desirable values in the piece-wise constant function.

Smiliar to Section 4 of Yun et al. (2019), Proposition 7 can be proved with five steps above, where the major difference here is in Step 1-3 we create **contextual mappings** for both node features and relative structural encodings. Next we explain the five steps in detail.

Step 1. Since $\mathcal{X}$ is bounded, we can assume without generality that $\mathcal{X} \subset (0,1)^d$. Thus, according to Lemma 5 in Yun et al. (2019), the input $\boldsymbol{X}$ can be quantized to grid $\mathbb{G}_\delta := \{0, \delta, \ldots, 1-\delta\}^{d\times n}$. We still use $\boldsymbol{x}_i$ to denote the quantized feature vector.

Step 2. Before this step, we have $l_j \in [0 : \delta : \delta^{-d+1} - \delta]$. Our goal in this step is to scale each l_j to $\delta^{-\delta^{-1}l_j}$. For every entry $\boldsymbol{L}_{:,j}$ in $\boldsymbol{L}$, the scaling function is defined as

$$\boldsymbol{L}_{:,j} \mapsto \boldsymbol{L}_{:,j} + (\delta^{-\delta^{-1}\boldsymbol{u}^\top \boldsymbol{L}_{:,j}} - \boldsymbol{u}^\top \boldsymbol{L}_{:,j})\boldsymbol{e}^{(1)}, \tag{25}$$

We use a group of feed-forward layers to approximate this function, which is possible because Transformer has residual connections. Note that after this process, $\{1, \delta^{-1}, \delta^{-2}, \ldots, \delta^{-\delta^{-d}+1}\}$ contains all possible values for l_j. As our proof can have δ arbitrarily small, we assume $\delta^{-1} > n$.

Step 3. Since $\mathcal{C}$ is finite we may assume $\mathcal{C} = \{1, 2, \ldots, c\}$, and let $\boldsymbol{W} = \{\boldsymbol{\alpha}_{r,s}\}_{r,s=1,\ldots,n}$. We use one self-attention layer consists of c attention heads to perform the desired global shift. We first define

$$\phi_i(\boldsymbol{W}) = \{\phi_i(\boldsymbol{\alpha}_{r,s})\}_{r,s=1,\ldots,n}, \tag{26}$$

$$\text{where } \phi_i(x) = \begin{cases} 1, \text{ if } x = i, \\ 0, \text{ else.} \end{cases} \tag{27}$$

Then, for $i = 1, 2, \ldots, c$, the i-th attention head is defined as

$$\psi_i(\boldsymbol{Z}) = \boldsymbol{e}^{(1)}\boldsymbol{u}^\top \boldsymbol{Z}\sigma_{\mathrm{H},\delta^{-p+1}/n!}(\phi_i(\boldsymbol{W})) \tag{28}$$

where $p = -\delta^{-d}$. Noticing $\lim_{\delta\to 0}\delta^{-p+1}/n! = 0$ and the fact in A.2.4, the selected $\sigma_{\mathrm{H},\delta^{-p+1}/n!}$ is acceptable. This $\phi_i(x)$ function can be learned by embedding layers operated on the relative structural encodings. And the final attention layer is computed as

$$\Psi(\boldsymbol{Z}) = \boldsymbol{Z} + \sum_{i=1}^{c} n!\delta^{(3p+q)i}\psi_i(\boldsymbol{Z}), \tag{29}$$

where q satisfies $\delta^{q+1} \leq n! \leq \delta^q$. Note that p and q are both negative. For the convenience of further description, we define $\boldsymbol{u}^\top\Psi(\boldsymbol{L}) = [\bar{l}_1, \ldots, \bar{l}_n]$.

Explanation on Step 2 and 3. We aim to generate the **bijective column id mapping** for each $\{\!\{(\boldsymbol{x}_j, \boldsymbol{\alpha}_{i,j}) : j = 1, \ldots, n\}\!\}$, while using only $\boldsymbol{X}$ as feature input and the structural encodings are leveraged by shift operations in Step 2 and 3. We further prove this in Proposition 7 below:

Proposition 7. *For any $u, v \in \{1, \ldots, n\}$, $\bar{l}_u = \bar{l}_v$ if and only if $\{\!\{(\boldsymbol{x}_j, \boldsymbol{\alpha}_{u,j}) : j = 1, \ldots, n\}\!\} = \{\!\{(\boldsymbol{x}_j, \boldsymbol{\alpha}_{v,j}) : j = 1, \ldots, n\}\!\}$, and every $\bar{l}_u$ is bounded.*

Proof. For each node u, we define $Y(u, i) = \{\!\{\boldsymbol{x}_v : \boldsymbol{\alpha}_{u,v} = i\}\!\}$ and $S(u, i) = \sum_{\boldsymbol{\alpha}_{u,v}=i} l_v$, where l_v is the scaled $\boldsymbol{u}^\top\boldsymbol{L}_{:,v}$ after Step 2.

Let the first row of $\psi_i(\boldsymbol{Z})$ be $[r_i(1), r_i(2), \ldots, r_i(n)]$. We first show that $r_i(u) = r_i(v)$ if and only if $Y(u, i) = Y(v, i)$. Due to the ingenious construction of $\boldsymbol{u}$ in Yun et al. (2019), l_j has been an injective descriptor of $\boldsymbol{x}_j$ before the scaling in Step 2. Since the scaling in Step 2 is injective, the scaled l_j also becomes an injective descriptor of $\boldsymbol{x}_j$. According to our scaling strategy, the scaled l_j can be viewed as a p-digit one-hot representation of $\boldsymbol{x}_j$. Noticing $\delta^{-1} > n$, $S(u, j)$, as the summation of these scaled l_j, also becomes a unique descriptor of $Y(u, i)$ and $1 \leq S(u, j) \leq \delta^p$.

Definition A.1. *Suppose the set of possible values of a is P. Then for any $u, v \in P$, if $|u - v|$ is always an integer multiple of s, then we call s the minimal distance between any unique choices of a.*

Accordingly, the minimal distance between any unique choices of $S(u, j)$ is 1 because the the minimal distance between any scaled l_j is 1.

Next, before discussing ψ_i in Equation (28), (29) and r_i, we first present a lamma:

Lemma 1. *For real numbers a, b, the minimal distance between any unique choices of b is s, and $a \leq m$ holds. $a + b$ becomes a unique descriptor of a if $2m < s$.*

Proof. Suppose we have $a_1 + b_1 = a_2 + b_2$ and $a_1 \neq a_2$. Then we have

$$|a_1 - a_2| = |b_1 - b_2|, \tag{30}$$

and $|b_1 - b_2| \geq s$, $|a_1 - a_2| \leq 2m$. Then the proof is completed by contradiction. □

Given the definition of ψ_i (please refer to Appendix B.5 in (Yun et al., 2019) for more details on the selective shift operation, which is the basis for the construction of ψ_i), we have

$$r_i(u) = (\frac{1}{k} + \frac{\delta^{-p+1}}{n!})S(u, i), \tag{31}$$

where $k = |Y(u, i)|$. According to the range of scaled l_j, we have

$$1 \leq \frac{1}{k}S(u, i) \leq \delta^{p+1}, \tag{32}$$

$$\frac{\delta^{-p+1}}{n!} \leq \frac{\delta^{-p+1}}{n!}S(u, i) \leq \frac{\delta}{n!}. \tag{33}$$

It is easy to infer that the minimal distance between any unique choices of $\frac{1}{k}S(u,i)$ is an integer multiple of $\frac{1}{n!}$, and we have

$$2 \cdot \frac{\delta}{n!} < 2 \cdot \frac{1}{2n!} = \frac{1}{n!}. \tag{34}$$

According to Lemma 1, $r_i(u)$ is a unique descriptor of $\frac{\delta^{-p+1}}{n!}S(u,i)$, then it is also a unique descriptor of $Y(u,j)$. Now we have $1 \leq r_i(u) \leq \delta^p$ and the minimal distance between any unique choices of $r_i(u)$ is $\frac{\delta^{-p+1}}{n!}$. The following Lemma is applied to the construction of Ψ:

Lemma 2. *For k positive real numbers $a_1, a_2, ..., a_k$, $\sum_{i=1}^{k} a_i$ is a unique descriptor of $(a_1, a_2, \ldots, a_k)$ if:*

1. *For any a_i, there exists r_i such that $a_i \leq r_i$,*
2. *Let $s(i)$ be the minimal distance between any unique choices of a_i, then $s(i) > \sum_{j=1}^{i-1} r_j$ holds for any i.*

Proof. Assuming that there exists two groups of positive real numbers $\{a_i^{(1)}\}_{i=1}^{k}$ and $\{a_i^{(2)}\}_{i=1}^{k}$ which both satisfy conditions above and $\sum_{i=1}^{k} a_i^{(1)} = \sum_{i=1}^{k} a_i^{(2)}$. Besides, the two group of numbers are not totally equal correspondingly, which means there must exist one $l \in \{1, 2, ..., c\}$ such that $a_l^{(1)} \neq a_l^{(2)}$ and for any $j > l$, $a_j^{(1)} = a_j^{(2)}$ holds.

According to the second condition, $|a_l^{(1)} - a_l^{(2)}| > \sum_{j=1}^{l-1} r_j$ holds. Since $\sum_{i=1}^{k} a_i^{(1)} = \sum_{i=1}^{k} a_i^{(2)}$, then

$$|a_l^{(1)} - a_l^{(2)}| = |\sum_{j=1}^{l-1} a_j^{(1)} - \sum_{j=1}^{l-1} a_j^{(2)}| \tag{35}$$

$$= |\sum_{j=1}^{l-1} (a_j^{(1)} - a_j^{(2)})| \tag{36}$$

$$\leq \sum_{j=1}^{l-1} |a_j^{(1)} - a_j^{(2)}| \tag{37}$$

$$\leq \sum_{j=1}^{l-1} r_j, \tag{38}$$

where the proof is completed by contradiction. □

Finally we consider the definition of Ψ. We have

$$\bar{l}_u = l_u + \sum_{i=1}^{c} n! \delta^{(3p+q)i} r_i(u), \tag{39}$$

and it can be concluded that

$$1 \leq l_u \leq \delta^{p+1}, \tag{40}$$

$$\delta^{3ip+(i+1)q+1} \leq n! \delta^{(3p+q)i} r_i(u) \leq \delta^{(3i+1)p+(i+1)q}, \text{ for } i = 1, \ldots, c. \tag{41}$$

And the minimal distance between any unique choices of $n! \delta^{(3p+q)i} r_i(u)$ is

$$s(i) = \delta^{(3i-1)p+iq+1}. \tag{42}$$

If $i = 1$, then $s(1) = \delta^{2p+q+1} > \delta^{p+1}$; if $i = j + 1$, then trivially we have

$$s(j+1) = \delta^{(3j+2)p+(j+1)q+1} > \sum_{s=1}^{j} \delta^{(3s+1)p+(s+1)q}. \tag{43}$$

Thus, according to the lemma above, $\bar{l}_u$ also becomes a unique descriptor of $\{\!\{Y(u,i) : i = 1, \ldots, n\}\!\}$, then it must be a unique descriptor of $\{\!\{(\boldsymbol{x}_u, \boldsymbol{\alpha}_{u,j}) : j = 1, \ldots, n\}\!\}$. We also have $\bar{l}_u$ bounded as $\bar{l}_u < \delta^{(3c+1)p+(c+1)q-1}$, which completes the proof. □

Step 4. After the previous steps, $\bar{l}_u$ is the unique id for $\{\!\{(\boldsymbol{x}_j, \boldsymbol{\alpha}_{u,j}) : j = 1, \ldots, n\}\!\}$, and we have $\bar{l}_u \in [0 : \delta : \delta^{(3c+1)p+(c+1)q-1} - \delta]$. It can be observed that if we define $d' = (3c+1)p+(c+1)q-1$ and treat d' as the "new" d, we can apply exactly the same methods in Appendix B.5 of Yun et al. (2019) to employ multiple selective shift operations and generate contextual embeddings for $\mathcal{H} = [\{\!\{(\boldsymbol{x}_j, \boldsymbol{\alpha}_{i,j}) : j = 1, \ldots, n\}\!\}]_{i=1,\ldots,n}$. Note that since we assume $\mathcal{X} \subset (0,1)^{d\times n}$, only Category 1 and 2 (Appendix B.5 of Yun et al. (2019)) need to be considered.

Step 5. Now with contextual embeddings $q(\mathcal{H})$, we can use methods in Appendix B.6 of Yun et al. (2019) to map every mapping values to the desired output computed by $\bar{f}$, which completes the proof. □

A.3 Proof for Theorem 3

Proof. Let the label mappings generated by S'-GT and S-GT at iteration t be g'_t and g_t respectively. We denote the conditions in Equation 7 and 8 as $f_A = p_A(f'_A)$ and $f_R = p_R(f'_R)$. For graphs $G_v = (V_v, E_v)$ and $G_u = (V_u, E_u)$ (G_v and G_u may be the same graph), we first show that for any node $v \in V_v$ and $u \in V_u$ at iteration t, if S'-GT generates $g'_t(v) = g'_t(u)$, then S-GT also gets $g_t(v) = g_t(u)$. For $t = 0$ this holds because if $g'_0(v) = g'_0(u)$ then v and u must have $h_0(v) = h_0(u)$ and $f'_A(v, G_v) = f'_A(u, G_u)$. Since $f_A = p_A(f'_A)$, it means that $f_A(v, G_v) = f_A(u, G_u)$, which leads to $g_0(v) = g_0(u)$. Suppose this condition holds for iteration $0, 1, \ldots, t$ and $g'_{t+1}(v) = g'_{t+1}(u)$. From the injectiveness of function Φ, we have

$$\{\!\{(g'_t(r), f'_R(v, r, G_v)) : r \in V_v\}\!\} = \{\!\{(g'_t(r), f'_R(u, r, G_u)) : r \in V_u\}\!\}. \tag{44}$$

If two finite multisets are identical, then the elements in the two multisets can be matched in pairs. The condition $f_R = p_R(f'_R)$ implies that for any r, s, $f'_R(v, r, G_v) = f'_R(u, s, G_u) \implies f_R(v, r, G_v) = f_R(u, s, G_u)$. Together with the assumption that $g'_t(v) = g'_t(u)$ implies $g_t(v) = g_t(u)$, we can conclude that

$$(g'_t(r), f'_R(v, r, G_v)) = (g'_t(s), f'_R(u, s, G_u)) \implies (g_t(r), f_R(v, r, G_v)) = (g_t(s), f_R(u, s, G_u)). \tag{45}$$

Therefore, we have

$$\{\!\{(g_t(r), f_R(v, r, G_v)) : r \in V_v\}\!\} = \{\!\{(g_t(r), f_R(u, r, G_u)) : r \in V_u\}\!\}, \tag{46}$$

which directly leads to $g_{t+1}(v) = g_{t+1}(u)$. Then the proposition above is proved by induction. Now that for any iteration t we have $g'_t(v) = g'_t(u) \implies g_t(v) = g_t(u)$, indicating that a mapping ψ_t exists such that for any node v, $g_t(v) = \psi_t(g'_t(v))$.

Now consider two graphs G_1 and G_2 where S-GT decides them as non-isomorphic after t iterations, then the multiset of all updated node labels $\{\!\{g_t(v) : v \in V\}\!\}$ must be different for two graphs. Since $\{\!\{g_t(v) : v \in V\}\!\} = \{\!\{\psi_t(g'_t(v)) : v \in V\}\!\}$, $\{\!\{g'_t(v) : v \in V\}\!\}$ must also be different for two graphs or we will reach a contradiction, which suggests that S'-GT distinguishes G_1 and G_2 after t iterations. □

A.4 Proof for Theorem 4

Proof. The formal definition for WL test is presented in Appendix B. Here we denote $N^+(v) = N(v) \cup \{v\}$ as the ego subgraph of node v. For label update of node v, the values of $\textit{Neighbor}_R$ divides the node set V into three parts: the central node v, the neighborhood nodes $N(v)$ and nodes out of v's ego subgraph $V \setminus N^+(v)$. Thus, the node label update function controlled by $\textit{Neighbor}_R$ can be viewed as

$$g_t(v) = \Phi(g_t(v), \{\!\{g_t(r) : r \in N(v)\}\!\}, \{\!\{g_t(s) : s \in V \setminus N^+(v)\}\!\}). \tag{47}$$

For the first part of the proof, we prove that *Neighbor*-GT can distinguish any non-isomorphic graphs distinguishable by WL test. We first show that for any node v, u at iteration t, if *Neighbor*-GT generates $g_t(v) = g_t(u)$, then WL will obtain $w_t(v) = w_t(u)$. For $t = 0$ this obviously holds. Suppose this condition holds for iteration $0, 1, \ldots, t$ and $g_{t+1}(v) = g_{t+1}(u)$. From the injectiveness

of function Φ, we have

$$(g_t(v), \{\!\{g_t(r) : r \in N(v)\}\!\}, \{\!\{g_t(s) : s \in V_v \setminus N^+(v)\}\!\}) \tag{48}$$

$$=(g_t(u), \{\!\{g_t(r) : r \in N(u)\}\!\}, \{\!\{g_t(s) : s \in V_u \setminus N^+(u)\}\!\}), \tag{49}$$

where V_v is the node set of graph that v belongs to, which is the same for V_u. Slicing the two equivalent tuples above will also get equivalent results, as

$$(g_t(v), \{\!\{g_t(r) : r \in N(v)\}\!\}) = (g_t(u), \{\!\{g_t(r) : r \in N(u)\}\!\}). \tag{50}$$

Therefore we have $w_{t+1}(v) = w_{t+1}(u)$, and the proposition above is proved by induction. Now that for any iteration t we have $g_t(v) = g_t(u) \implies w_t(v) = w_t(u)$, indicating that a mapping ψ_t exists such that for any node v, $w_t(v) = \psi_t(g_t(v))$.

Consider two graphs G_1 and G_2 where WL decides them as non-isomorphic after t iterations, then the multiset of all updated node labels $\{\!\{w_t(v) : v \in V\}\!\}$ must be different for two graphs. Since $\{\!\{w_t(v) : v \in V\}\!\} = \{\!\{\psi_t(g_t(v)) : v \in V\}\!\}$, $\{\!\{g_t(v) : v \in V\}\!\}$ must also be different for two graphs, which suggests that *Neighbor*-GT can distinguish G_1 and G_2 after t iterations.

In the second part of the proof we only need to show that any non-isomorphic graphs indistinguishable by WL test can not be distinguished by *Neighbor*-GT. Suppose there are two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ that WL test cannot distinguish and the iteration converges at iteration t. Then for any $v, u \in V_1$, $w_t(v) = w_t(u)$ implies $w_{t+1}(v) = w_{t+1}(u)$ (the same for V_2), and there exists a bijective mapping $\theta : V_1 \to V_2$ such that for any $v \in V_1$, $w_t(v) = w_t(\theta(v))$ and $w_{t+1}(v) = w_{t+1}(\theta(v))$. Since w_t can be viewed as an absolute structural encoding function, we denote *Neighbor*$^+ = (w_t, \textit{Neighbor}_R)$ and *Neighbor*$^+$ must be more powerful than *Neighbor* according to Theorem 3 because *Neighbor*$^+ \succeq$ *Neighbor*. Let g^+ be the label mapping generated by *Neighbor*$^+$-GT on G_1 and G_2, and we may assume without generality that $g_0^+ = w_t$. For node $v \in V_1$, its first updated label is computed by

$$g_1^+(v) = \Phi(w_t(v), \{\!\{w_t(r) : r \in N(v)\}\!\}, \{\!\{w_t(s) : s \in V_1 \setminus N^+(v)\}\!\}). \tag{51}$$

Consider $v, u \in V_1$ where $w_t(v) = w_t(u)$. According to the definition of WL test, $w_{t+1}(v) = w_{t+1}(u)$ implies $\{\!\{w_t(r) : r \in N(v)\}\!\} = \{\!\{w_t(r) : r \in N(u)\}\!\}$. And because v and u belongs to the same graph, we also have $\{\!\{w_t(s) : s \in V_1 \setminus N^+(v)\}\!\} = \{\!\{w_t(s) : s \in V_1 \setminus N^+(u)\}\!\}$. This results in $g_1^+(v) = g_1^+(u)$.

Next we consider $v \in V_1$ and $\theta(v) \in V_2$ where $w_t(v) = w_t(\theta(v))$ and $w_{t+1}(v) = w_{t+1}(\theta(v))$. According to the definition of WL test, $w_{t+1}(v) = w_{t+1}(\theta(v))$ implies $\{\!\{w_t(r) : r \in N(v)\}\!\} = \{\!\{w_t(r) : r \in N(\theta(v))\}\!\}$. Since WL test can not distinguish G_1 and G_2, we have $\{\!\{w_t(s) : s \in V_1\}\!\} = \{\!\{w_t(s) : s \in V_2\}\!\}$, indicating that $\{\!\{w_t(s) : s \in V_1 \setminus N^+(v)\}\!\} = \{\!\{w_t(s) : s \in V_2 \setminus N^+(\theta(v))\}\!\}$, which shows $g_1^+(v) = g_1^+(\theta(v))$.

Together with statements above, for any $v, u \in V_1$ with $g_0^+(v) = g_0^+(u)$, we have $g_1^+(v) = g_1^+(u)$ and $g_1^+(v) = g_1^+(\theta(v))$. As θ is a bijective mapping, we can conclude that a mapping μ exists such that for any $v \in V_1 \cup V_2$, $g_1^+(v) = \mu(g_0^+(v))$, which tells us that $\{\!\{g_1^+(s) : s \in V_1\}\!\} = \{\!\{g_1^+(s) : s \in V_2\}\!\}$ and *Neighbor*$^+$-GT has not update any useful information in its first iteration. Therefore, we can see that $\{\!\{g_t^+(s) : s \in V_1\}\!\} = \{\!\{g_t^+(s) : s \in V_2\}\!\}$ for any t by induction, then *Neighbor*$^+$-GT can not distinguish G_1 and G_2. Because *Neighbor*$^+$-GT is more powerful than *Neighbor*-GT, *Neighbor*-GT also can not distinguish the two graphs, meaning that any non-isomorphic graphs indistinguishable by WL test can not be distinguished by *Neighbor*-GT, which completes the proof. □

A.5 Proof for Theorem 5

Proof. We can easily show that *SPD*-GT is more powerful than *Neighbor*-GT using Theorem 3 since two nodes are linked if there shortest path distance is 1. And according to Theorem 4, *Neighbor*-GT is as powerful as WL, then *SPD*-GT is more powerful than WL.

Figure 2 below shows a pair of graphs that can be distinguished by *SPD*-GT but not WL, which completes the proof. □

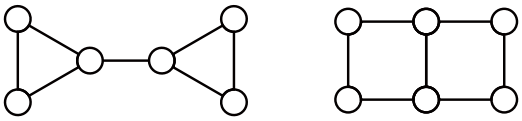


Figure 2: Two graphs that can be distinguished by *SPD*-GT but not WL.

A.6 Proof for Proposition 1

Proof. Let C_l denote the cycle graph of length l. Then consider two graphs G_1 and G_2, where G_1 consists of $2k+4$ identical C_{2k+3} graphs, and G_2 consists of $2k+3$ identical C_{2k+4} graphs. G_1 and G_2 have the same number of nodes, and the induced k-hop neighborhood of any node in either of the two graphs is simply a path of length $2k+1$. As a result, for structural encoding scheme S with k-hop receptive field, S-GT generates identical labels for every node in the two graphs, making G_1 and G_2 indistinguishable for S-GT. However, in G_2 there exists shortest paths of length $k+2$ while G_1 not, so *SPD*-GT can distinguish the two graphs. □

A.7 Proof for Theorem 6

Proof. Considering $\textit{SPD}_R$ is the first dimension of $\textit{SPIS}_R$, we have *SPIS* $\succeq$ *SPD* and we can prove *SPIS*-GT is more powerful than *SPD*-GT according to Theorem 3.

Figure 3 below shows a pair of graphs that can be distinguished by *SPIS*-GT but not *SPD*-GT. It is trivial to verify that *SPD*-GT can not distinguish them. For *SPIS*-GT, to understand this, Figure 3 colors examples of SPIS between non-adjacent nodes in the two graphs, where the nodes at two endpoints are colored as red. In the first graph, every SPIS between non-adjacent nodes has 3 nodes, but in the second graph there exists SPIS between non-adjacent nodes that has 4 nodes, so *SPIS*-GT can distinguish them. □

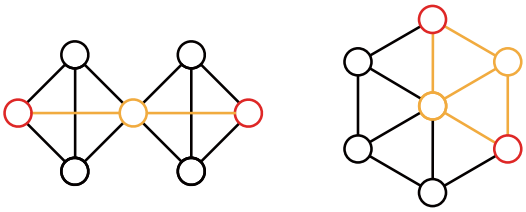


Figure 3: Two graphs that can be distinguished by *SPIS*-GT but not *SPD*-GT.

A.8 Proof for Proposition 2

Proof. It is trivial to verify that regular graphs with different parameters can be distinguished by WL, so we focus on strongly regular graphs with the same n and k but different λ and μ. For $\text{SRG}(n, k, \lambda, \mu)$, since every non-adjacent pair of nodes has μ neighbors in common, the SPIS between evry non-adjacent pair of nodes will have $\mu+2$ nodes, which implies that *SPIS*-GT can distinguish strongly regular graphs with different n, k, μ. Besides, the four parameters of strongly regular graphs are not independent, they satisfy

$$\lambda = k - 1 - \frac{\mu}{k}(n-k-1), \tag{52}$$

so *SPIS*-GT can distinguish strongly regular graphs with different parameters. □

A.9 Proof for Proposition 3

Proof. Figure 4 below shows a pair of graphs that can be distinguished by *SPIS*-GT but not 3-WL. The two graphs, named as the Shrikhande graph and the Rook's 4×4 graph, are both $\text{SRG}(16, 6, 2, 2)$

and the most popular example for indistinguishability with 3-WL (Arvind et al., 2020). To show they can be distinguished by *SPIS*-GT, Figure 4 also colors examples of SPIS between non-adjacent nodes, where the nodes at two endpoints are colored as red. In the second graph (the Shrikhande graph), one can verify that every SPIS between non-adjacent nodes has 4 nodes and 4 edges, but in the first graph (the Rook's 4×4 graph) there exists SPIS between non-adjacent nodes that has 5 edges, making *SPIS*-GT capable of distinguishing them. □

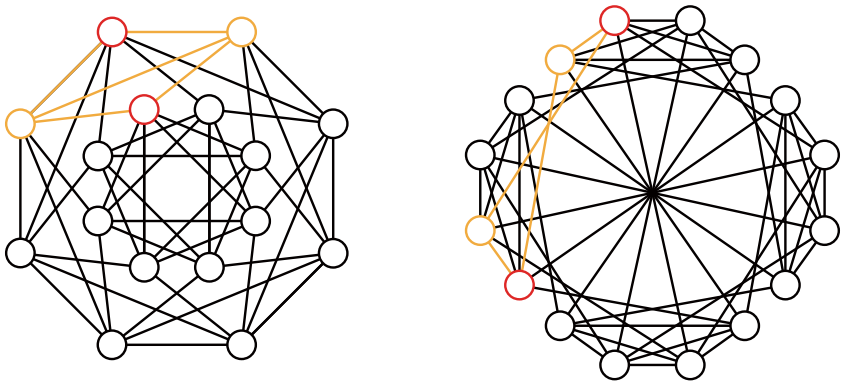


Figure 4: Two graphs (the Shrikhande graph and the Rook's 4×4 graph) that can be distinguished by *SPIS*-GT but not 3-WL.

B Related Work

Weisfeiler-Lehman Graph Isomorphism Test. The Weisfeiler-Lehman test is a hierarchy of graph isomorphism tests (Weisfeiler & Leman, 1968; Grohe, 2017), and the 1-WL test is know to be an upper bound on the expressivity of message-passing GNNs (Xu et al., 2018). Note that in this paper, without further notations, we will use the term *WL* to refer to 1-WL test. Formally, the definition of WL test is presented as

Definition B.1 (WL test)**.** *Let the input be a labeled graph $G = (V, E)$ with label map $h_0 : V \to \mathcal{X}$. WL test iteratively updates node labels of G, where at the t-th iteration, the updated node label map $w_t : V \to \mathcal{X}$ is computed as*

$$w_t(v) = \Phi\left(w_{t-1}(v), \{\!\{w_{t-1}(u) : u \in \mathcal{N}(v)\}\!\}\right), \tag{53}$$

where $w_0 = h_0$ and Φ is a function that injectively maps the collection of all possible tuples in the r.h.s. of Equation 53 to $\mathcal{X}$. We say two graphs G_1, G_2 are distinguished as non-isomorphic by WL test if after t iterations, the WL test generates $\{\!\{w_t(v)|v \in V_1\}\!\} \neq \{\!\{w_t(v)|v \in V_2\}\!\}$ for some t.

GNNs beyond the Expressivity of 1-WL. Since standard GNNs (like GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017) and GIN (Xu et al., 2018)) have expressive power bounded by the 1-WL, many works have proposed to improve the expressivity of GNNs beyond the 1-WL. High-order GNNs including (Morris et al., 2019; Maron et al., 2019; Azizian & Lelarge, 2020; Morris et al., 2020b) build graph neural networks inspired from k-WL with $k > 3$ to acquire the stronger expressive power, yet they mostly have high computational costs and complex network designs. Some works have proposed to use pre-computed topological node features to enhance the expressive power of GNNs, including (Monti et al., 2018; Liu et al., 2020; Bouritsas et al., 2022). These additional features may contain the number of the appearance of certain substructures like triangles, rings and circles. And recent works like (You et al., 2021; Vignac et al., 2020; Sato et al., 2021; Wijesinghe & Wang, 2021) show that the expressivity of GNNs can also be enhanced using random node identifiers or improved message-passing schemes.

The Transformer Architecture. Transformer is first proposed in Vaswani et al. (2017) to model sequence-to-sequence functions on text data, and now has become the prevalent neural architecture for natural language processing (Devlin et al., 2018). A Transformer layer mainly consists of a

multi-head self-attention (MHA) module and a position-wise feed-forward network (FFN) with residual connections. For queries $\boldsymbol{Q} \in \mathbb{R}^{n_q \times d}$, keys $\boldsymbol{K} \in \mathbb{R}^{n_k \times d}$ and values $\boldsymbol{V} \in \mathbb{R}^{n_k \times d}$, the scaled dot-product attention module can be defined as

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\boldsymbol{A})\boldsymbol{V}, \boldsymbol{A} = \frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}, \tag{54}$$

where n_q, n_k are number of elements in queries and keys, and d is the hidden dimension. Then, the multi-head attention is calculated as

$$\text{MHA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\boldsymbol{W}^O, \tag{55}$$

$$\text{head}_i = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V), \text{for } i = 1, \ldots, h, \tag{56}$$

where h is number of attention heads, $\boldsymbol{W}_i^Q \in \mathbb{R}^{d \times d_k}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d \times d_k}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d \times d_v}$ and $\boldsymbol{W}^O \in \mathbb{R}^{hd_v \times d}$ are projection parameter matrices, d, d_k, d_v are the dimension of hidden layers, keys and values. In encoder side of the original Transformer architecture, all queries, keys and values come from the input sequence embeddings.

After multi-head attention, the position-wise feed-forward network is applied to every element in the sequence individually and identically. This network is composed of two linear transformations, an activation function and residual connections in between. Layer normalization Ba et al. (2016) is also performed before the multi-head self-attention and feed-forward network Xiong et al. (2020). A Transformer layer can be defined as below:

$$\text{Transformer}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{FFN}(\text{LN}(\boldsymbol{H})) + \boldsymbol{H}, \tag{57}$$

$$\boldsymbol{H} = \text{MHA}(\text{LN}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})) + \boldsymbol{Q}. \tag{58}$$

Graph Transformers. Along with the recent surge of Transformer, many prior works have attempted to bring Transformer architecture to the graph domain, including GT Dwivedi & Bresson (2020), GROVER Rong et al. (2020), Graphormer Ying et al. (2021), SAN Kreuzer et al. (2021), Gophormer Zhao et al. (2021), GraphGPS (Rampášek et al., 2022) and SAT (Chen et al., 2022). These methods generally treat input graph as a sequence of node features, and apply various methods to inject structural information into the network. GT Dwivedi & Bresson (2020) provides a generalization of Transformer architecture for graphs with modifications like using Laplacian eigenvectors as positional encodings and adding edge feature representation to the model. GROVER Rong et al. (2020) is a molecular large-scale pretrain model that applies Transformer to node embeddings calculated by GNN layers. Graphormer Ying et al. (2021) proposes an enhanced Transformer with centrality, spatial and edge encodings, and achieves state-of-the-art performance on many molecular graph representation learning benchmarks. SAN Kreuzer et al. (2021) presents a learned positional encoding that cooperates with full Laplacian spectrum to learn the position of each node in the graph. Gophormer Zhao et al. (2021) applies structural-enhanced Transformer to sampled ego-graphs to improve node classification performance and scalability. GraphGPS (Rampášek et al., 2022) proposes a recipe on how to build a general, powerful, scalable (GPS) graph Transformer with linear complexity and state-of-the-art results on real benchmark tests. SAT (Chen et al., 2022) proposes the Structure-Aware Transformer with its new self-attention mechanism which incorporates structural information into the original self-attention by extracting a subgraph representation rooted at each node using GNNs before computing the attention.

C More Discussions

Isomorphic Structural Encodings. For any structural encoding function, we say f_A is *isomorphic* to f'_A if there exists a bijective mapping p such that $f_A = p(f'_A)$, which is the same for f_R. It is trivial to conclude that isomorphic structural encodings have the same expressive power.

Reduction of Absolute Structural Encodings. It can be observed that for structural encoding scheme $S = (f_A, f_R)$, f_A and f_R may express overlapping information and can be reduced to form a more concise representation. Since the principal phase of GT test is the label update controlled by relative structural encodings, we focus on the case where f_A can be deduced from f_R, and we can reduce f_A to eliminate redundant information, which is defined as

Definition C.1 (Reduction of Absolute Structural Encodings)**.** *A structural encoding scheme $S = (f_A, f_R)$ can be reduced to $S' = (id_A, f_R)$ if there exists mapping p such that for any $G = (V, E)$ and $v \in V$ we have*

$$f_A(v, G) = p(\{\!\{f_R(v, u, G) : u \in V\}\!\}) \tag{59}$$

Proposition 8. *If structural encoding scheme S can be reduced to S', then two graphs can be distinguished by S-GT if and only if they are distinguishable by S'-GT.*

Proof. According to Theorem 3, S-GT is more powerful than S'-GT, thus we only need to prove that any graphs distinguishable by S-GT can be distinguished by S'-GT. Let the label mappings generated by S'-GT and S-GT at iteration t be g'_t and g_t respectively. For graphs $G_v = (V_v, E_v)$ and $G_u = (V_u, E_u)$ (G_v and G_u may be the same graph), we first show that for any node $v \in V_v$ and $u \in V_u$ at iteration t, if S'-GT generates $g_{t+1}(v) = g_{t+1}(u)$, then S-GT also gets $g_t(v) = g_t(u)$. For $t = 0$, from the injectiveness of Φ we have

$$\{\!\{(g'_0(r), f'_R(v, r, G_v)) : r \in V_v\}\!\} = \{\!\{(g'_0(r), f'_R(u, r, G_u)) : r \in V_u\}\!\}. \tag{60}$$

Accordingly, we have

$$\{\!\{f'_R(v, r, G_v) : r \in V_v\}\!\} = \{\!\{f'_R(u, r, G_u) : r \in V_u\}\!\}, \tag{61}$$

which directly leads to $f_A(v, G_v) = f_A(u, G_u)$. According to the definition of S', we have $g_0(v) = g_0(u)$. Suppose this condition holds for iteration $0, \ldots, t$ and $g'_{t+1}(v) = g'_{t+1}(u)$. From the injectiveness of function Φ, we have

$$\{\!\{(g'_t(r), f_R(v, r, G_v)) : r \in V_v\}\!\} = \{\!\{(g'_t(r), f_R(u, r, G_u)) : r \in V_u\}\!\}. \tag{62}$$

According to the assumption that $g'_t(v) = g'_t(u)$ implies $g_{t-1}(v) = g_{t-1}(u)$, we can infer that

$$\{\!\{(g_{t-1}(r), f_R(v, r, G_v)) : r \in V_v\}\!\} = \{\!\{(g_{t-1}(r), f_R(u, r, G_u)) : r \in V_u\}\!\}, \tag{63}$$

which directly leads to $g_t(v) = g_t(u)$. Then the proposition above is proved by induction. Now that for any iteration t we have $g'_{t+1}(v) = g'_{t+1}(u) \implies g_t(v) = g_t(u)$, indicating that a mapping ψ_t exists such that for any node v, $g_t(v) = \psi_t(g'_{t+1}(v))$.

Now consider two graphs G_1 and G_2 where S-GT decides them as non-isomorphic after t iterations, then the multiset of all updated node labels $\{\!\{g_t(v) : v \in V\}\!\}$ must be different for two graphs. Since $\{\!\{g_t(v) : v \in V\}\!\} = \{\!\{\psi_t(g'_{t+1}(v)) : v \in V\}\!\}$, $\{\!\{g'_{t+1}(v) : v \in V\}\!\}$ must also be different for two graphs or we will reach a contradiction, which suggests that S'-GT distinguishes G_1 and G_2 after $t + 1$ iterations. □

The proposition above guarantees that the reduction of redundant encodings will not influence the expressive power of corresponding GT test. For example, since the degree of nodes can be obtained by counting its neighbors, then $(\textit{Deg}_A, \textit{Neighbor}_R)$ can be reduced to $(\textit{id}_A, \textit{Neighbor}_R)$.

Computing *SPIS*. To compute *SPIS* encoding on a input graph $G = (V, E)$, we first use the Floyd-Warshall algorithm (Floyd, 1962) to compute the lengths of shortest paths between all pairs of vertices in G, which takes $O(n^3)$ time complexity where $n = |V|$. Next for every pair of nodes (v, u), for every node s we test if s is in $\text{SPIS}(v, u)$ by checking if $\text{SPD}_R(v, s) + \text{SPD}_R(s, u) = \text{SPD}_R(v, u)$ holds to construct $V_{\text{SPIS}(v,u)}$, and this step also has $O(n^3)$ time complexity. Finally, for every pair of nodes (v, u) we construct $E_{\text{SPIS}(v,u)}$ by computing the intersection between $V_{\text{SPIS}(v,u)} \times V_{\text{SPIS}(v,u)}$ and E. If we denote the average number of nodes of SPISs in the graph as t, then $V_{\text{SPIS}(v,u)} \times V_{\text{SPIS}(v,u)}$ can have t^2 edges in average and thus the final step costs $O(n^2t^2)$ complexity. The overall time complexity for computing *SPIS* is then $O(n^3 + n^2t^2)$. As we can reasonably expect $t^2 \sim n$ on most real-world sparse graphs because SPISs should be small with respect to the entire graph, the complexity of *SPIS* can be viewed as $O(n^3)$. This is quite acceptable because the time complexity for computing *SPD* via Floyd-Warshall algorithm is already $O(n^3)$, and *SPIS* offers a much stronger expressive power.

D Connections between GT Test and Previous Graph Transformers

As we have discussed above, GT test is capable of characterizing the expressive power of most graph Transformers, and here we will present some examples. Note that in the scope of this paper, we only consider simple undirected graphs with node features.

Graphormer (Ying et al., 2021). The Graphormer model utilizes three types of structural encodings: *Centrality Encoding* that encodes node degrees, *Spatial Encoding* that encodes the structural relation between nodes via shortest path distance, and *Edge Encoding* that captures information of edges that connect two nodes (which we do not consider since it relates to edge feature). The *Centrality Encoding* corresponds to the $\textit{Deg}_A$ absolute structural encoding we discuss in Section 4.1, and the *Spatial Encoding* is equivalent to the shortest path distance encoding $\textit{SPD}_R$ in Section 4.1. Therefore, similar to the proof for Proposition 4, we can prove that the expressivity of Graphormer with two types of structural encoding above can be characterized with *Graphormer*-GT, where

$$\textit{Graphormer} = (\textit{Deg}_A, \textit{SPD}_R). \tag{64}$$

According to Proposition 8, the *Graphormer* encoding above can be reduced to $\textit{SPD} = (\textit{id}_A, \textit{SPD}_R)$ since the degree of node v can be inferred from the number of node v such that $\textit{SPD}(v, u) = 1$. Thus, the expressivity of Graphormer can be characterized with *SPD*-GT. According to our analysis in Section 5.1, *SPD*-GT is strictly more powerful than WL and has unique expressive power elaborated by Proposition 1.

GT (Dwivedi & Bresson, 2020) and SAN (Kreuzer et al., 2021). GT and SAN both employ Laplacian eigenvalues and eigenvectors as absolute structural encodings, and during Transformer layers the embedding update strategy is determined by link connections. For both models, it can be easily verified that $\textit{Laplacian}^k_A$ below characterizes their absolute structural encodings:

$$\textit{Laplacian}^k_A(v, G) = (\Lambda^k_G, \lambda^k_v), \tag{65}$$

where Λ^k_G is the k smallest Laplacian eigenvalues of graph G, λ^k_v is the Laplacian eigenvector of v in G corresponding to Λ^k_G, and every $\textit{Laplacian}^k_A(v, G)$ comes from a *deterministic factorization policy for graph Laplacian matrix*. As for relative structural encoding, since during Transformer layers both models only consider if two nodes are linked, we can conclude that $\textit{Neighbor}_R$ summarizes the expressivity of embedding update process. Therefore, $\textit{Laplacian}^k$-GT is an upper bound on the expressivity of SAN and GT model, where

$$\textit{Laplacian}^k = (\textit{Laplacian}^k_A, \textit{Neighbor}_R). \tag{66}$$

It is quite difficult to accurately analyze the expressive power of $\textit{Laplacian}^k$ since it relates to the sign invariance of Laplacian eigenvectors and contents of spectral graph theory. However, since $\textit{Laplacian}^k$ only involves the $\textit{Neighbor}_R$ relative encoding, our Theorem 4 shows that for SAN and GT, the exploitation of Transformer network results in no improvement on the structural expressive power when comparing with GNNs using $\textit{Laplacian}^k_A$ as additional node features.

Gophormer (Zhao et al., 2021). Gophormer is a scalable graph Transformer model for node classification with proximity-enhanced multi-head attention (PE-MHA) as the core module for learning graph structure. When analyzing the structural expressive power of Gophormer, the global nodes added to represent global information are ignored. It can be concluded that the following $\textit{Proximity}^k_R$ relative structural encoding characterizes the expressivity of PE-MHA in Gophormer:

$$\textit{Proximity}^k_R(v_i, u_j, G) = (\boldsymbol{I}(i, j), \tilde{\boldsymbol{A}}(i, j), \ldots, \tilde{\boldsymbol{A}}^k(i, j)), \tag{67}$$

where $\boldsymbol{I}$ is the identity matrix, and $\tilde{\boldsymbol{A}} = \text{Norm}(\boldsymbol{A} + \boldsymbol{I})$ is the normalized adjacency matrix with self-loop. Since Gophormer employs no absolute structural encoding, $\textit{Proximity}^k$-GT describes the expressivity of Gophormer, where $\textit{Proximity}^k = (\textit{id}_A, \textit{Proximity}^k_R)$.

As for any v_i, v_j, $\boldsymbol{A}(i, j)$ can be inferred from $(\boldsymbol{I}(i, j), \tilde{\boldsymbol{A}}(i, j))$, the $\textit{Proximity}^k$ structural encoding is more expressive than *Neighbor* when $k \leq 1$. As a result, according to Theorem 4, $\textit{Proximity}^k$-GT is more powerful than WL, and one can easily verify that two graphs in Figure 2 can be distinguished by $\textit{Proximity}^k$-GT. Therefore, we can conclude that Gophormer with $\textit{Proximity}^k$ encoding is strictly more powerful than WL.

SAT (Chen et al., 2022) SAT propose the Structure-Aware Transformer with its new self-attention mechanism which incorporates structural information into the original self-attention by extracting a subgraph representation rooted at each node using GNNs before computing the attention. Theoretical results in the SAT paper guarantees that SAT is at least as expressive as the GNN subgraph extractor,

and using GT test we will arrive at the similar result. In the framework of GT test, regardless of absolute structural encoding, SAT model incorporates the node features generated by GNNs as relative structural encoding at each structure-aware attention:

$$SAT_R^{\text{subtree}}(v,u,G) = (\text{GNN}_G^{(k)}(v), \text{GNN}_G^{(k)}(u)) \text{ (k-subtree GNN extractor)}, \tag{68}$$

$$SAT_R^{\text{subgraph}}(v,u,G) = (\sum_{u\in N_k(v)} \text{GNN}_G^{(k)}(u), \sum_{r\in N_k(u)} \text{GNN}_G^{(k)}(r)) \text{ (k-subgraph GNN extractor)}. \tag{69}$$

For k-subtree GNN extractor, considering that $\text{GNN}_G^{(k)}(v)$ can be inferred from $\{\{SAT_R^{\text{subtree}}(v,u,G) : u \in V\}\}$ by choosing the first element of each tuple, with proposition 8 we can conclude that $\text{SAT}^{\text{subtree}}$ can be viewed as having absolute structural encoding generated by GNN_G^k, which is the same for k-subgraph GNN extractor. Therefore, SAT^{subtree}-GT is more powerful than $\phi(v,G) = \text{GNN}_G^{(k)}(v)$, and SAT^{subgraph}-GT is more powerful than $\phi(v,G) = \sum_{u\in N_k(v)} \text{GNN}_G^{(k)}(u)$, which shows that the expressivity upper bound of SAT is more powerful than its GNN feature extractor.

E Graph Representation Learning Experiments

E.1 Datasets

Datasets	#Graphs	#Nodes	#Node Attributes	#Edges	#Edge Attributes	#Tasks
ZINC(subset)	12,000	277920	1	597960	1	1
QM9	130831	2359210	11	4883516	4	12
QM8	21786	169339	79	352356	10	16
ESOL	1128	14991	9	15428	3	1

Table 3: Statics for graph regression datasets.

Datasets	#Graphs	#Nodes	#Node Attributes	#Edges	#Edge Attributes	#Classes
PTC-MR	344	4015	18	10108	4	2
MUTAG	188	3371	7	7442	4	2
COX2	467	19252	35	40578	-	2
PROTEINS	1113	43471	3	162088	-	2

Table 4: Statics for graph classification datasets.

Statistics of the datasets used in this work are summarized in Table 3 and 4. The ZINC dataset from benchmarking-gnn (Dwivedi et al., 2020)[4] is a subset of the ZINC chemical database (Irwin & Shoichet, 2005) with 12000 molecules, and the task is to predict the solubility of molecules. We follow the guidelines and use the predefined split for training, validation and testing. QM9 and QM8 (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014; 2015) are two molecular datasets containing small organic molecules up to 9 and 8 heavy atoms, and the task is to predict molecular properties calculated with ab initio Density Functional Theory (DFT). We follow the guidelines in MoleculeNet (Wu et al., 2018) for choosing regression tasks and metrics. We perform joint training on 12 tasks for QM9 and 16 tasks for QM8. ESOL is also a molecular regression dataset in MoleculeNet containing water solubility data for compounds[5]. PTC-MR, MUTAG, COX2 and PROTEINS are four graph classification datasets collected from TUDataset (Morris et al., 2020a)[6]. On graph regression datasets, We use random 8:1:1 split for training, validation, and testing except for ZINC, and report the performance averaged over 3 runs. On graph classification datasets, we use 10-fold cross validation with 90% training and 10% testing, and report the mean best accuracy.

[4] `https://github.com/graphdeeplearning/benchmarking-gnns`.

[5] QM8, QM9 and ESOL are available at `http://moleculenet.ai/datasets-1` (MIT 2.0 license).

[6] The four datasets are available at `https://chrsmrrs.github.io/datasets/`.

E.2 SETTINGS

E.2.1 GRAPHORMER VARIANTS

Model Description. In graph representation learning experiments, We use four Graphormer variants based on four structural encoding schemes discussed in the main paper: *SPIS*, *SPD*, *Neighbor* and *id*. For Graphormer-*SPIS*, to incorporate the extra structural information encoded by *SPIS* encoding while not making significant changes to the model architecture, we replace the spatial encoding $b_{SPD_R(v,u,G)}$ in Graphormer with $b_{SPD_R(v,u,G)} + \text{Linear}(|V_{SPIS(v,u)}|, |E_{SPIS(v,u)}|)$, and keep the remaining network components unchanged. Graphormer-*SPD* is basically the original Graphormer architecture. In Graphormer-*Neighbor*, we remove the edge encoding since it contains information beyond the neighborhood connections, and replace the spatial encoding $b_{SPD_R(v,u,G)}$ in Graphormer with $b_{Neighbor_R(v,u,G)}$. Similarly, for Graphormer-*id*, we remove the centrality encoding and edge encoding, and substitute the spatial encoding $b_{SPD_R(v,u,G)}$ in Graphormer with $b_{id_R(v,u,G)}$.

	ZINC	QM9	QM8	ESOL	PTC-MR	MUTAG	COX2	PROTEINS
peak_learning_rate	2e-4	3e-4	3e-4	5e-4	0.01	0.01	0.01	0.01
end_learning_rate	1e-9	1e-9	1e-9	1e-9	1e-9	1e-9	1e-9	1e-9
hidden_dim	80	512	256	256	256	256	256	256
ffn_dim	80	512	256	256	256	256	256	256
weight_decay	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0
input_dropout_rate	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
attention_dropout_rate	0.1	0.0	0.1	0.1	0.1	0.1	0.1	0.1
dropout_rate	0.1	0.0	0.1	0.1	0.1	0.1	0.1	0.1
num_layers	12	20	6	16	16	16	16	16
num_heads	8	32	16	16	16	16	16	16

Table 5: Model configurations and hyper-parameters of Graphormer with different types of structural encoding.

Model Configurations. We report the detailed hyper-parameter settings used for training the Graphormer variants in Table 5. We use the source code provided by (Ying et al., 2021) (MIT 2.0 license) and use AdamW (Loshchilov & Hutter, 2018) as optimizer and linear decay as learning rate scheduler. All models are trained on 2 NVIDIA RTX 3090 GPUs for up to 12 hours.

E.2.2 BASELINES

	ZINC	QM9	QM8	ESOL	PTC-MR	MUTAG	COX2	PROTEINS
peak_learning_rate	3e-4	3e-4	3e-4	1e-3	3e-4	3e-4	3e-4	3e-4
end_learning_rate	1e-9	1e-9	1e-5	1e-9	1e-9	1e-9	1e-9	1e-9
hidden_dim	256	256	256	512	256	256	256	256
weight_decay	0.01	0.01	0.0	0.01	0.01	0.01	0.01	0.01
input_dropout_rate	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0
dropout_rate	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
num_layers	16	16	16	5	5	5	5	5
num_heads(only for GAT)	4	4	4	4	4	4	4	4

Table 6: Model configurations and hyper-parameters of GNN baselines.

Model Configurations. We report the detailed hyper-parameter settings used for training GNN baselines including GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), GIN (Xu et al., 2018) and GraphSAGE (Hamilton et al., 2018) in Table 6. During training stage, we use AdamW (Loshchilov & Hutter, 2018) as optimizer and decay the learning rate with a cosine annealing utilized in (Loshchilov & Hutter, 2016). All models are trained on 2 NVIDIA RTX 3090 GPUs until convergence for up to 12 hours.

For SAT (Chen et al., 2022) model, it has substantially higher complexity than all proposed methods and baselines with its GNN-based feature extractor. We follow the instructions and run the code in

`https://github.com/BorgwardtLab/SAT` on ZINC dataset. Due to limitations on computational resources, to give a fair comparison, we run the model for 3.5 days with almost 1000 epochs, and report the best performance.

E.3 PERFORMANCES ON QM9

Task	Unit	MAE			
		Graphormer-*id*	Graphormer-*Neighbor*	Graphormer-*SPD*	Graphormer-*SPIS*
μ	D	8.1654±0.1095	0.6926±1.646e-4	0.3688±3.010e-4	0.3536±3.727e-4
α	a_0^3	24.562±0.1815	0.8597±6.886e-4	0.2417±8.542e-7	0.2365±1.105e-3
ϵ_{HOMO}	eV	1.5222±0.0283	0.1962±3.667e-4	0.0683±2.186e-5	0.0664±5.848e-5
ϵ_{LUMO}	eV	4.3868±0.2717	0.2644±5.850e-5	0.0699±1.036e-5	0.0686±9.445e-5
$\Delta\epsilon$	eV	0.6235±0.0126	0.3407±4.459e-4	0.0933±1.420e-4	0.0904±2.811e-4
$\langle R^2 \rangle$	a_0^2	166.64±12.339	76.885±2.309e-2	18.774±7.047e-2	18.174±3.046e-2
ZPVE	eV	1.3654±0.0391	0.0165±3.954e-6	0.0061±2.012e-4	0.0055±5.311e-7
U_0	eV	3457.2±274.96	1.0558±8.925e-4	3.8210±7.458e-2	2.1069±3.581e-4
U	eV	2041.3±47.641	1.0552±2.932e-4	3.8882±2.049e-1	2.1069±3.694e-4
H	eV	3593.4±31.424	1.0540±6.737e-4	3.7888±1.232e-1	2.1007±4.798e-4
G	eV	1468.9±97.816	1.0505±6.409e-4	3.8175±1.508e-1	2.0994±3.115e-4
c_{v}	$\frac{\text{cal}}{\text{mol K}}$	5.4585±0.1456	0.4510±1.725e-4	0.1034±5.555e-5	0.1027±6.856e-7

Table 7: Performance on QM9, reported by separate tasks.

Here we additionally report the performance of Graphormer variants over 12 tasks individually on QM9 dataset in Table 7.

F LIMITATIONS AND POSSIBLE NEGATIVE SOCIETAL IMPACTS

Limitations. It is well-known that self-attention in Transformer network has quadratic complexity with respect to the input size, and since GT test is proposed to characterize the expressivity of graph Transformers, it inherits this complexity issue and each label update iteration of GT test costs $O(n^2)$ complexity (equivalent to 2-WL), where n is the input graph size. Besides, the structural encodings may be computed by algorithms with relative high complexity, like *SPD* which is obtained by the $O(n^3)$ Floyd-Warshall algorithm, and in Appendix C we formulate the complexity of proposed *SPIS* as $O(n^3 + n^2t^2)$. Still, we believe it is worth studying the expressive power of graph Transformers despite these limitations on complexity. It is shown that the global receptive field brought by self-attention can lead to higher performance than traditional GNNs on real-world benchmarks (Ying et al., 2021). Additionally, as Transformer gain popularity in multiple areas of machine learning, the complexity issue of Transformers can be mostly resolved by low-complexity self-attention techniques and modern computational devices specially optimized for Transformers (like NVIDIA's H100 GPU). Therefore, together with our theoretical results which show graph Transformers can exhibit outstanding expressive power, we believe Transformers will be widely used in graph machine learning due to their performance and expressivity, despite their higher complexity than GNNs.

Ethic Statement and Possible Negative Societal Impacts. This work is a foundational research on the expressivity of graph Transformers and is not tied to any particular applications. Therefore, our work may have potential negative societal impacts with malicious use of graph neural models (like generating fake profiles) or environmental impact (like training huge graph Transformers).