

Article

A Data-Driven Graph Neural Network Framework for Predicting Topological Indices of Unicyclic and Bicyclic Graphs

Nadia Khan ¹, Muhammad Zeeshan ², Yousaf Iqbal ³, Mansoor Iqbal ⁴, Muhammad Amjad Iqbal ^{4,5} and Sheraz Aslam ^{5,6,*}

¹ School of Computing Sciences, Pak-Austria Fachhochschule IAST, Haripur 22621, Pakistan; nadia.khan@paf-iaast.edu.pk

² Mathematics and Statistics Department, Eastern Michigan University, Ypsilanti, MI 48197, USA; mzeeshan@emich.edu

³ Department of Physics, University of Poonch, Rawalakot 12350, Pakistan; yousafiqbal@upr.edu.pk

⁴ Eratosthenes Center of Excellence, 3012 Limassol, Cyprus; mansoor.iqbal@eratosthenes.org.cy (M.I.); muhammadamjad.iqbal@eratosthenes.org.cy (M.A.I.)

⁵ Department of Computer Science, CTL Eurocollege, 3077 Limassol, Cyprus

⁶ Department of Computer Science, American University of Cyprus, 6019 Larnaca, Cyprus

* Correspondence: aslam.sheraz@aucy.ac.cy

Abstract

Topological indices provide numerical descriptions of graph structure and support graph analysis in cheminformatics, network design, and graph mining. This study presents a reproducible computational framework that combines controlled cyclic-graph generation, structure-preserving transformations, exact computation of six classical topological indices, and multi-output graph neural network regression. The framework evaluates the Wiener, Merrifield–Simmons, Hosoya, first Zagreb, second Zagreb, and Randić indices for unicyclic and bicyclic graphs. It represents each graph using sparse connectivity and node-level features that encode degree, cycle membership, pendant connectivity, leaf status, and normalized eccentricity. A graph isomorphism network (GIN) jointly predicts the six indices and is compared with graph convolutional networks (GCNs), graph attention networks (GATs), and descriptor-based regression baselines. The controlled benchmark shows that nonlinear descriptor-based models achieve the lowest aggregate errors because the supplied graph-level descriptors contain strong prior information about graph size, degree structure, branching, and cycle complexity. Although GIN does not achieve the highest overall accuracy, it provides the strongest graph-native performance by learning directly from sparse connectivity and node-level features without requiring a fixed handcrafted graph-level descriptor vector. The proposed surrogate does not replace exact evaluation for isolated small graphs, where exact computation remains more appropriate. Instead, its practical value emerges through repeated evaluations of larger, more complex graph instances. To examine this setting, a computational stress experiment evaluates sparse multicyclic graphs under increasing cyclomatic complexity and measures exact computation time, timeout frequency, prediction accuracy, and the amortized break-even point. The results indicate that surrogate prediction becomes beneficial when combinatorial index computation becomes sufficiently expensive, and the trained model is reused across many structurally related graph queries. An external experiment on circulant graphs also demonstrates that the framework can extend beyond the original graph generators by modifying only the graph-construction stage.



Academic Editor: Ronghua Li

Received: 24 June 2026

Revised: 23 July 2026

Accepted: 3 August 2026

Published: 11 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article

distributed under the terms and

conditions of the [Creative Commons](#)

[Attribution \(CC BY\)](#) license.

Keywords: graph neural networks; topological indices; unicyclic graphs; bicyclic graphs; synthetic graph generation; multi-target regression; graph representation learning

1. Introduction

In network analysis, biology, chemistry, and communication systems, graphs offer a natural way to depict organized data. In these domains, graph-level descriptors facilitate effective structure comparison and downstream prediction, whereas vertices and edges record entities and their interactions. Because topological indices simplify a graph to numerical numbers that encapsulate degree-based properties, branching structure, and distance patterns, they are among the most used descriptors. Thus, in graph-based modeling and analysis, classical indices like the Wiener, Merrifield–Simmons, Hosoya, first Zagreb, second Zagreb, and Randić indices continue to be significant.

Unicyclic and bicyclic graphs provide controlled families for studying how cyclic structure affects classical topological indices. For a connected graph with n vertices and m edges, they satisfy $m = n$ and $m = n + 1$, respectively, and occur in ring-based molecular and restricted-connectivity systems. Exact computation remains efficient for small isolated graphs, particularly for degree-based indices and the Wiener index; therefore, the proposed framework is not intended to replace deterministic evaluation in such cases. Instead, it separates offline exact labeling from reusable online prediction for repeated analysis of structurally related graphs, especially when the Merrifield–Simmons and Hosoya indices become costly because they require counting independent sets and matchings. The bounded benchmark validates the complete workflow under controlled conditions, while the stress experiment examines when surrogate prediction becomes computationally justified as exact combinatorial cost increases. Accordingly, the study should be interpreted as a benchmark-oriented investigation of the conditions under which surrogate prediction becomes computationally justified, rather than as a general replacement for deterministic graph algorithms.

There are two primary approaches used by current research on this issue. One approach uses exact formulas, extremal results, and structural inequalities to investigate topological indices on unicyclic and bicyclic graphs. For graph-level regression and classification tasks, particularly in molecular and network applications, another approach creates graph neural networks [1,2].

These directions rarely intersect in a framework that treats multiple classical topological indices as prediction targets on cyclic graph families generated under explicit structural rules and transformations [3]. This study fills that gap by constructing a synthetic benchmark, defining an exact labeling pipeline, and evaluating multi-output graph neural networks against conventional regression baselines. Figure 1 presents a conceptual overview of the proposed framework.

Earlier studies have combined controlled structural generation with machine-learning analysis for unicyclic graphs, circulant topologies, and synthetic combinational circuits [4]. The present study does not claim that synthetic graph generation or invariant prediction represents an entirely new idea. Instead, it contributes a unified benchmark focused on unicyclic and bicyclic graphs. The benchmark integrates structure-preserving transformations, exact computation of six classical indices, reproducible data partitioning, descriptor-based regression, graph-native multi-output learning, runtime analysis, and external-family evaluation. Accordingly, the circulant experiment serves as a portability assessment rather than as a contribution to a new circulant dataset. Overall, the study provides a systematic bench-

mark for comparing exact computation, descriptor-based regression, and graph-native multi-target learning across structured cyclic graph families.

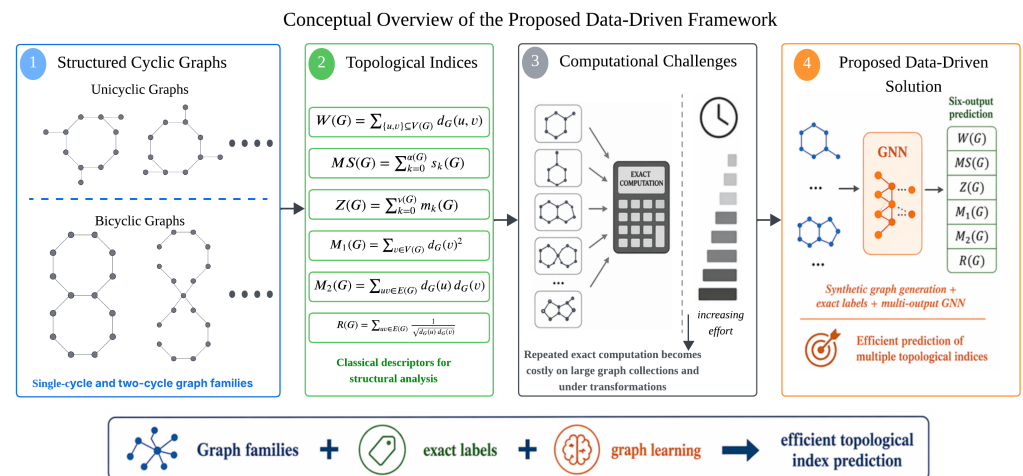


Figure 1. Conceptual overview of the proposed framework for topological index prediction on unicyclic and bicyclic graphs.

Research Gap and Contributions

Current literature does not provide a unified and reproducible framework that combines controlled graph generation, structure-preserving transformations, exact computation of multiple classical topological indices, graph-native multi-output prediction, and an explicit analysis of when surrogate inference becomes computationally worthwhile. Most graph-theoretic studies derive exact results for individual indices or constrained graph families, whereas many data-driven studies use topological indices as precomputed descriptors rather than prediction targets. A further unresolved issue concerns computational justification: surrogate inference has limited value when exact evaluation is inexpensive, deterministic, and required only once. A valid assessment must therefore consider the growth of exact-computation cost, prediction accuracy, one-time training cost, deployment volume, and the corresponding amortized break-even point.

This study addresses this gap through the following contributions:

- It formalizes controlled generators for unicyclic and bicyclic graphs, together with local transformations that preserve graph simplicity, connectivity, and cyclomatic number.
- It develops a reproducible exact-labeling pipeline for the Wiener, Merrifield–Simmons, Hosoya, first Zagreb, second Zagreb, and Randić indices.
- It constructs a deduplicated graph benchmark with explicit train–validation–test partitioning and compares graph-native and descriptor-based multi-output regression models under identical data partitions, input-scaling procedures, and evaluation metrics.
- It distinguishes overall predictive accuracy from graph-native representation learning. Descriptor-based nonlinear models achieve the lowest aggregate errors when informative handcrafted summaries are available, whereas GIN provides the strongest graph-native performance without requiring a fixed graph-level descriptor vector.
- It introduces a larger sparse-multicyclic stress experiment that quantifies exact-computation growth, timeout frequency, predictive performance, inference cost, and amortized break-even behavior as graph order and cyclomatic number increase.
- It evaluates external-family transferability using circulant graphs and shows that the graph-source module can be replaced while preserving the downstream labeling, representation, training, and evaluation workflow.

Together, these contributions establish a reproducible comparative framework for exact labeling, descriptor-based regression, and graph-native prediction, rather than an architecture-specific superiority claim.

The remainder of this paper is structured as follows: Section 2 reviews related work on conventional graph theory, computational methods, and ML techniques. Section 3 describes the exact computation of indices, the data-generation pipeline, and the architecture of the proposed learning model. Section 4 outlines the measurement procedures and experimental conditions. In Section 5, results are analyzed through comparisons of the proposed model and ML-based methods in terms of numerical accuracy and the efficiency of the proposed approach. Finally, Section 6 concludes the paper and discusses directions for future research.

2. Related Work

The current work has a solid mathematical foundation thanks to research on topological indices of unicyclic and bicyclic graphs. For indices like the Wiener, Merrifield–Simmons, Hosoya, Zagreb, and Randić on bounded classes of cyclic graphs, existing work in this field yields accurate formulas, extremal values, and structural bounds [5–10]. Graphs that maximize or reduce particular indices under constraints on graph order, size, degree sequence, diameter, or number of pendant vertices are described in a variety of publications [11]. These findings shed light on how branching patterns, cycle configuration, and local structure affect index values. Because it encourages the choice of unicyclic and bicyclic graph families and supports the use of accurate index calculation as the labeling stage of the suggested framework, this line of research is directly relevant to the current methodology.

Algorithmic analysis in applied graph domains is also supported by topological indices [12]. These descriptors are employed in quantitative structure–property and structure–activity studies in mathematical chemistry, where ring-based compounds and fused cyclic systems are represented by molecular graphs [13,14]. Research in the field establishes links between these indices and biological activity, molecular stability, and the physical–chemical behavior of the substance [15,16]. The study of ring and dual-ring structures in communication networks, where cyclic connectivity affects redundancy, routing scheme topology, and load distribution, can rely on descriptors that correspond to those of network science [17]. In applications where cyclic graphs are generated iteratively and cyclic transformations occur locally, this highlights the importance of efficient evaluation and motivates the use of cyclic graphs [18].

Circulant graphs are relevant to communication-system design because their regular structure, short path lengths, and scalable routing properties support networks-on-chip and multicore interconnection systems [19–23]. These studies examine ring-based, two-dimensional, multiplicative, shortest-path, and deadlock-free routing formulations and provide practical motivation for evaluating cyclic and circulant graph structures.

While that work focuses on optimal circulant-topology synthesis, the present study reconstructs circulant graphs to test whether the exact-labeling and graph-learning pipeline developed for unicyclic and bicyclic graphs can be retrained for six classical topological indices. This direction also aligns with controlled synthetic graph-generation studies for machine-learning analysis, such as labeled combinational-circuit datasets [24,25].

Another direction involves graph learning approaches that emphasize graph neural networks, particularly for graph-level prediction problems. Initial research developed methods such as convolutional networks and message passing to build representations from neighboring nodes at both the vertex and graph levels [26]. Subsequent work developed graph convolutional networks, graph attention networks, and graph isomorphism networks and applied them to molecular property prediction, graph classification, and regression on structured graph data [27]. Several contributions also examined the expressive

power of GNNs and their relation to graph isomorphism tests, especially through comparisons with the Weisfeiler–Leman framework [28,29]. These studies are closely aligned in methodological approach to that presented in this paper, as the proposed prediction phase uses graph-level representation learning and considers a variety of GNN architectures for multi-output regression tasks.

The expressive power of message-passing graph neural networks is also relevant to the present prediction task. Standard message-passing architectures are bounded by the discriminative power of the first-order Weisfeiler–Leman (1-WL) test. With injective neighborhood aggregation and readout functions, GIN can attain the distinguishing power of 1-WL. However, it cannot generally distinguish graphs that remain equivalent under 1-WL refinement [28,29]. Therefore, strong graph-level discrimination within this framework does not imply that the model can represent every global graph invariant exactly.

This limitation is particularly relevant to the Merrifield–Simmons and Hosoya indices. The quantity $MS(G)$ counts all independent vertex sets, whereas $Z(G)$ counts all matchings. Both are global combinatorial counting functions whose exact computation is intractable in general. Graphs that a 1-WL-bounded architecture cannot distinguish may still differ in their global substructure counts and related combinatorial invariants. Therefore, a conventional message-passing GNN has no general guarantee of representing $MS(G)$ or $Z(G)$ exactly, even when trained using exact labels. Distribution-specific approximation remains possible for bounded graph families, but such empirical approximation should not be interpreted as an exact combinatorial representation.

Recent work based on topology-aware GNNs has shown the importance of structural information for GNNs [1,30]. The present study takes the topology-enabled concept a step further by applying it to graph-level regression, predicting multiple global indices from sparse graph structure and node-level topological features. Another body of research focuses on the ability of graph learning models to approximate graph invariants and structural properties [31,32]. Previous literature has examined the possibility of estimating spectral properties, classical graph invariants, and other structural properties using GNNs [33,34]. However, related applications typically treat topological indices as precomputed scalars and use them as input features for subsequent prediction tasks [35]. This distinction is important in the context of the current study. The proposed method in this paper does not require topological indices as input features. Instead, it generates a synthetic, labeled dataset via combinatorial calculations and trains GNNs to predict six indices simultaneously.

There is still a gap between the exact index analysis and graph-based learning. Classical studies yield exact results on graph-like data (structured graphs), while graph learning provides flexible predictive models for graph-level tasks. The synthesis of cyclic graphs has been explored in the context of structure-preserving transformations, multiple indices for exact labeling, sparse graph representation, and multi-output regression, with few isolated frameworks being offered in the context of a reproducible computational pipeline. This study is intended to fill this void, providing a benchmark-oriented methodological framework rather than a precise graph-theoretic calculation.

3. Methodology

The methodology pipeline employed in this work involves five steps: formal definition of graph families and the desired graph invariants; construction of graphs through structural transformations; calculation of topological indices for these graphs; representation of graphs as input to the supervised learning algorithm; and the regression problem using graphs and a graph neural network combined with the baselines. These steps are expressed

operationally for clarity and to be replicated. The overall computational workflow of the proposed framework is summarized in Figure 2.

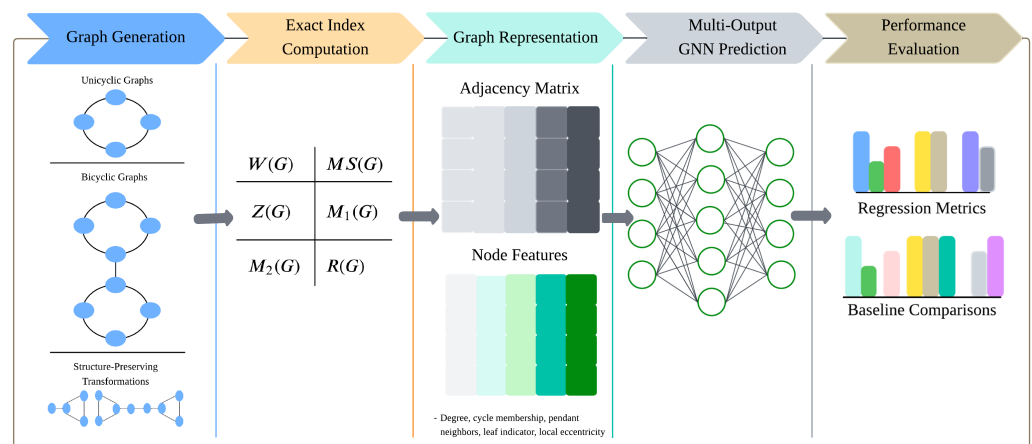


Figure 2. Overview of the proposed pipeline for topological index prediction on unicyclic and bicyclic graphs.

3.1. Mathematical Framework

Let $G = (V(G), E(G))$ be a finite, simple, connected graph with $n = |V(G)|$ vertices and $m = |E(G)|$ edges. Its cyclomatic number is defined as

$$\mu(G) = m - n + 1. \quad (1)$$

Graph

G is unicyclic when $\mu(G) = 1$, equivalently $m = n$, and bicyclic when $\mu(G) = 2$, equivalently $m = n + 1$. These definitions refer to the number of independent cycles, as measured by the dimension of the cycle space, rather than to the total number of distinct simple cycles. The connectivity condition is essential; for example, a cycle together with an isolated vertex is not unicyclic. The study considers only simple connected graphs without loops or multiple edges.

Figure 3 illustrates representative examples of the two graph families considered in this study.

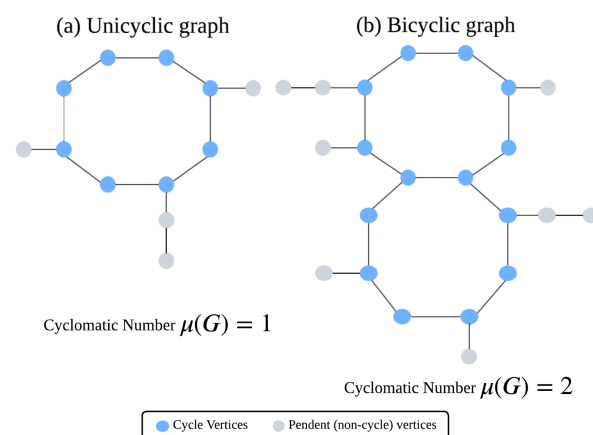


Figure 3. Representative examples of the unicyclic and bicyclic graph families considered in this study.

The predictive task targets six classical topological indices. The study computes each index exactly and uses the resulting values as regression targets.

- The Wiener index:

$$W(G) = \sum_{\{u,v\} \subseteq V(G)} d_G(u,v), \quad (2)$$

where the sum runs over all unordered pairs of distinct vertices.

- The Merrifield–Simmons index:

$$MS(G) = \sum_{k=0}^{\alpha(G)} s_k(G), \quad (3)$$

where $s_k(G)$ is the number of independent vertex sets of size k , and $\alpha(G)$ is the independence number of G .

- The Hosoya index:

$$Z(G) = \sum_{k=0}^{\nu(G)} m_k(G), \quad (4)$$

where $m_k(G)$ is the number of matchings of size k , and $\nu(G)$ is the matching number of G .

- The first Zagreb index:

$$M_1(G) = \sum_{v \in V(G)} d_G(v)^2. \quad (5)$$

- The second Zagreb index:

$$M_2(G) = \sum_{uv \in E(G)} d_G(u) d_G(v). \quad (6)$$

- The Randić index:

$$R(G) = \sum_{uv \in E(G)} \frac{1}{\sqrt{d_G(u) d_G(v)}}. \quad (7)$$

The methodology also considers local graph transformations that preserve the graph family. Let $\mathcal{G}_1$ denote the unicyclic family and $\mathcal{G}_2$ denote the bicyclic family generated in this study. Each transformation modifies a local adjacency pattern while preserving graph simplicity, connectivity, and cyclomatic number.

For unicyclic graphs, the study applies two transformation types.

- Pendant-edge relocation: Let $G \in \mathcal{G}_1$ contain a unique cycle C . Select a pendant edge xy such that $x \in V(C)$, and y is a leaf. Select a second vertex $z \in V(C)$ such that $z \neq x$, and $zy \notin E(G)$. Remove xy and add zy . The transformed graph remains simple, connected, and unicyclic.
- Pendant-subtree reattachment: Let T be a pendant tree attached at a cycle vertex x . Select a second cycle vertex $z \neq x$ such that reattaching T at z does not create parallel edges. Remove the attachment edge at x and attach the root of T to z . The transformed graph remains simple, connected, and unicyclic.

For bicyclic graphs, the study applies analogous operations around the bicyclic core.

- Pendant relocation on the bicyclic core: The procedure reattaches a pendant edge or a pendant subtree from one vertex of the bicyclic core to another non-adjacent admissible core vertex.
- Core-preserving rewiring: The procedure rewires an edge adjacent to a pendant structure while verifying that the resulting graph remains simple, connected, and satisfies $\mu(G) = 2$.

The transformation routine accepts a transformed graph only when the following conditions hold simultaneously: the graph is simple, the graph is connected, and the cyclomatic number remains unchanged. Any potential transformations that do not meet any of these requirements are eliminated by the implementation. Representative structure-preserving transformations for unicyclic and bicyclic graphs, such as pendant-edge relocation and pendant-subtree reattachment under fixed cyclomatic number, are shown in Figure 4.

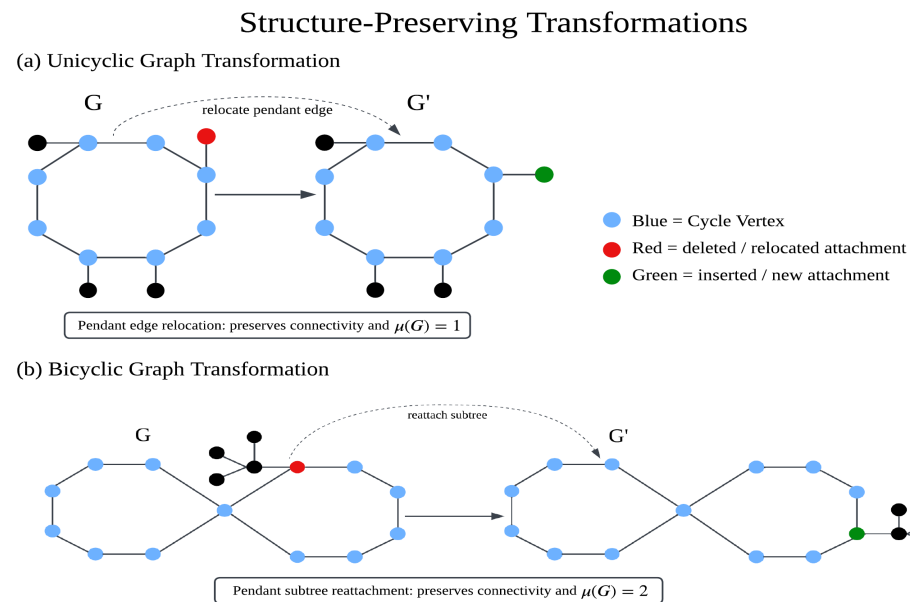


Figure 4. Structure-preserving transformations on unicyclic and bicyclic graphs.

3.2. Synthetic Graph Generation

No standard public benchmark currently provides labeled unicyclic and bicyclic graphs with exact values of the six target indices under explicit structure-preserving transformations [4]. The study, therefore, constructs a synthetic benchmark with controlled graph generation, exact labeling, duplicate removal, and stratified evaluation splits. This offline-generation and online-prediction design is methodologically related to synthetic combinational-circuit generation, where controlled instances are constructed, evaluated, and subsequently modeled using machine learning [25].

The construction process generates graphs of controlled order, applies valid local transformations, removes duplicate structures, and computes exact labels for each retained graph.

The generator samples the graph order from the interval

$$n \in \{6, 7, \dots, 30\}.$$

The lower bound excludes degenerate cases and ensures that both graph families admit non-trivial pendant substructures, defined here as at least one vertex or attached tree lying outside the cyclic core and connected to it through a single attachment edge. The upper bound preserves the tractability of the combinatorial index computations. For each admissible order n , the generator produces N_u unicyclic graphs and N_b bicyclic graphs. In the default experimental configuration, the study sets

$$N_u = 200 \quad \text{and} \quad N_b = 200$$

per order. This configuration yields an initial pool of $25 \times 200 = 5000$ unicyclic graphs and 5000 bicyclic graphs before transformation-based augmentation.

The unicyclic generator proceeds in four steps.

1. Sample a cycle length ℓ uniformly from $\{3, 4, \dots, n\}$.
2. Construct the cycle C_ℓ .
3. Add the remaining $n - \ell$ vertices one at a time.
4. For each added vertex, select an attachment vertex uniformly from the set of current vertices and insert one edge.

This construction yields a connected simple graph with exactly one cycle because each new vertex contributes one edge and increases the vertex count by one.

The bicyclic generator proceeds in five steps.

1. Sample a bicyclic core type from the set {vertex-sharing, edge-sharing, bridge-connected}.
2. Sample admissible cycle lengths ℓ_1 and ℓ_2 subject to the selected core type and the target graph order n .
3. Construct a simple connected bicyclic core satisfying $\mu(G) = |E(G)| - |V(G)| + 1 = 2$.
4. Add the remaining vertices one at a time as pendant-tree extensions attached to existing vertices. Each such addition preserves the cyclomatic number because it introduces one new vertex and one new edge.
5. Reject any generated graph that violates simplicity, connectivity, or the condition $\mu(G) = 2$.

This construction yields connected simple graphs with cyclomatic number 2. The implementation verifies $\mu(G) = 2$ after each generation step.

The generation process also applies local transformations to enrich structural diversity. For each retained graph, the algorithm samples up to $T = 2$ valid transformations. The procedure stores the original graph and each transformed graph only when graph-isomorphism testing confirms that the new graph does not duplicate a previously stored instance. The implementation uses canonical graph labeling to remove duplicates before index computation.

Figure 5 summarizes the synthetic graph generation and labeling pipeline used to construct the benchmark dataset.

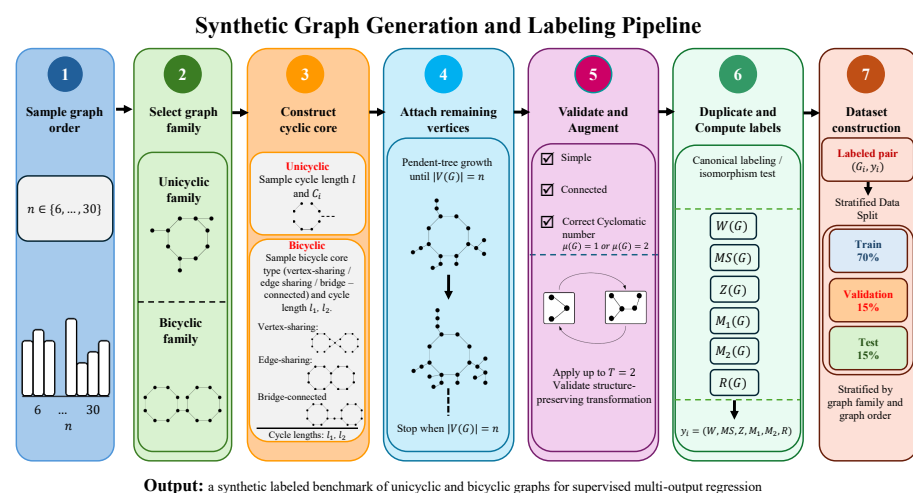


Figure 5. Synthetic graph generation and labeling pipeline used to construct the benchmark dataset.

Algorithm 1 summarizes the full generation pipeline.

The final dataset consists of labeled graph–target pairs

$$(G_i, \mathbf{y}_i), \quad \mathbf{y}_i = (W(G_i), MS(G_i), Z(G_i), M_1(G_i), M_2(G_i), R(G_i)). \quad (8)$$

The split ratio is 70% for training, 15% for validation, and 15% for testing. The split procedure stratifies jointly by graph family and graph order so that each subset preserves comparable structural distributions.

Algorithm 1 Synthetic Graph Generation and Labeling Pipeline

Require: Graph orders $\{6, 7, \dots, 30\}$, number of unicyclic graphs per order N_u , number of bicyclic graphs per order N_b , number of transformations per graph T

Ensure: Labeled dataset $\mathcal{D}$ with graph-index pairs and train/validation/test splits

```

1: Initialize empty dataset  $\mathcal{D}$ 
2: for  $n = 6$  to  $30$  do
3:   for  $i = 1$  to  $N_u$  do
4:     Generate a unicyclic graph  $G \in \mathcal{G}_1$  of order  $n$ 
5:     if  $G$  is not isomorphic to any graph already stored in  $\mathcal{D}$  then
6:       Compute exact target vector  $\mathbf{y}(G)$ 
7:       Store  $(G, \mathbf{y}(G))$  in  $\mathcal{D}$ 
8:       for  $t = 1$  to  $T$  do
9:         Generate a valid transformed graph  $G'$ 
10:        if  $G'$  is simple, connected, unicyclic, and new then
11:          Compute exact target vector  $\mathbf{y}(G')$ 
12:          Store  $(G', \mathbf{y}(G'))$  in  $\mathcal{D}$ 
13:        end if
14:      end for
15:    end if
16:  end for
17:  for  $i = 1$  to  $N_b$  do
18:    Generate a bicyclic graph  $G \in \mathcal{G}_2$  of order  $n$ 
19:    if  $G$  is not isomorphic to any graph already stored in  $\mathcal{D}$  then
20:      Compute exact target vector  $\mathbf{y}(G)$ 
21:      Store  $(G, \mathbf{y}(G))$  in  $\mathcal{D}$ 
22:      for  $t = 1$  to  $T$  do
23:        Generate a valid transformed graph  $G'$ 
24:        if  $G'$  is simple, connected, bicyclic, and new then
25:          Compute exact target vector  $\mathbf{y}(G')$ 
26:          Store  $(G', \mathbf{y}(G'))$  in  $\mathcal{D}$ 
27:        end if
28:      end for
29:    end if
30:  end for
31: end for
32: Stratify  $\mathcal{D}$  by graph family and graph order
33: Split  $\mathcal{D}$  into training, validation, and test sets
34: return  $\mathcal{D}$ 

```

3.3. Exact Computation of Topological Indices

The study computes each target index exactly before model training. This stage provides reliable labels and separates target generation from statistical estimation.

The Wiener index requires all-pairs shortest-path distances. Since all graphs are unweighted, the implementation runs the breadth-first search from each vertex. For a graph with n vertices and m edges, this procedure has time complexity $O(n(n + m))$.

The first and second Zagreb indices and the Randić index are degree-based quantities. The implementation computes the degree sequence once, then traverses the vertex set or edge set directly. The corresponding time complexity is $O(n + m)$.

The Merrifield–Simmons index requires independent set enumeration, while the Hosoya index calls for the enumeration of matchings. Combinatorial enumeration is involved, and hence, a recursive divide-and-conquer algorithm using memoization is used

on subgraph instances. This is still feasible since we are considering graphs having up to 30 nodes and exhibiting a sparse cycle connectivity. The intermediate instances are stored using signature values of induced subgraphs.

Algorithm 2 summarizes the labeling stage.

Algorithm 2 Exact Computation of Six Topological Indices

Require: A connected simple graph G

Ensure: Exact target vector $\mathbf{y}(G) = (W(G), MS(G), Z(G), M_1(G), M_2(G), R(G))$

- 1: Compute all-pairs shortest-path distances in G by repeated breadth-first search
 - 2: Compute the Wiener index $W(G)$ from the distance matrix
 - 3: Compute the degree sequence $\{d_G(v) : v \in V(G)\}$
 - 4: Compute the first Zagreb index $M_1(G)$
 - 5: Compute the second Zagreb index $M_2(G)$
 - 6: Compute the Randić index $R(G)$
 - 7: Count independent vertex sets with memoization and compute the Merrifield–Simmons index $MS(G)$
 - 8: Count matchings with memoization and compute the Hosoya index $Z(G)$
 - 9: Form the target vector $\mathbf{y}(G)$
 - 10: **return** $\mathbf{y}(G)$
-

3.4. Graph Representation and Input Features

Each graph is represented in PyTorch Geometric using a sparse edge-index tensor $\mathbf{E} \in \mathbb{N}^{2 \times 2m}$ and a node-feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{feat}}}$, where n and m denote the numbers of vertices and undirected edges, respectively. The tensor $\mathbf{E}$ stores both directions of every undirected edge. Each graph retains its original number of vertices and edges; therefore, the representation requires no padding, truncation, fixed-size adjacency matrices, or node masks. During mini-batch processing, PyTorch Geometric combines variable-size graphs using a batch-assignment vector by the following:

- Normalized vertex degree;
- Binary cycle-membership indicator;
- Number of pendant neighbors;
- Binary leaf indicator;
- Local eccentricity normalized by graph order.

The cycle-membership indicator equals one when the vertex belongs to at least one cycle and equals zero otherwise. The implementation obtains cycle membership during graph generation and verifies it through cycle-basis computation. The local eccentricity of a vertex is the maximum shortest-path distance from that vertex to any other vertex in the graph.

The implementation standardizes continuous node features by the mean and standard deviation computed from the training set only. The model does not use target values in feature construction.

3.5. Predictive Model Architecture

The predictive component uses a graph isomorphism network (GIN) as the primary graph neural network architecture because GIN models provide strong discriminative power for graph-level learning tasks and operate effectively on sparse graphs. The study also evaluates graph convolutional networks (GCNs) and graph attention networks (GATs) as comparative GNN architectures in the experimental section.

Figure 6 illustrates the architecture of the primary GIN model used for multi-output prediction.

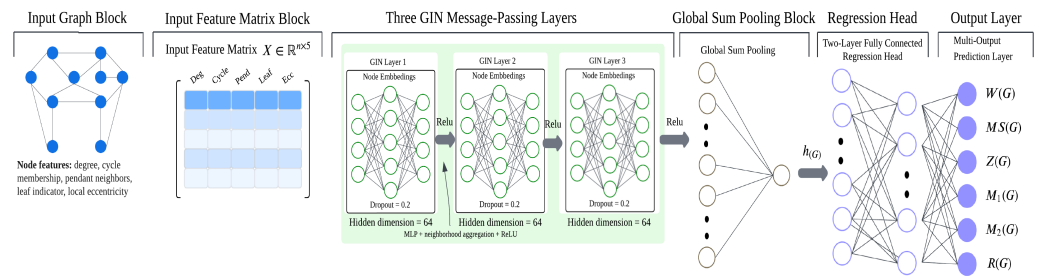


Figure 6. Architecture of the primary GIN model for multi-output prediction of six topological indices.

The primary GIN model contains three message-passing layers. Let $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ denote the initial feature vector of vertex v . Each GIN layer updates node features according to

$$\mathbf{h}_v^{(\ell)} = \text{MLP}^{(\ell)} \left((1 + \epsilon^{(\ell)}) \mathbf{h}_v^{(\ell-1)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(\ell-1)} \right), \quad (9)$$

where $\epsilon^{(\ell)}$ is a learnable scalar, $\mathcal{N}(v)$ is the neighbourhood of v , and $\text{MLP}^{(\ell)}$ is a two-layer multilayer perceptron with ReLU activation.

The hidden dimension equals 64 in each message-passing layer. The architecture applies batch normalization and dropout with rate 0.2 after each layer. After the third message-passing layer, the model aggregates node embeddings by global sum pooling:

$$\mathbf{h}_G = \sum_{v \in V(G)} \mathbf{h}_v^{(3)}. \quad (10)$$

A two-layer fully connected regression head then maps $\mathbf{h}_G$ to a six-dimensional prediction vector

$$\hat{\mathbf{y}} = f_{\text{out}}(\mathbf{h}_G) \in \mathbb{R}^6. \quad (11)$$

The model predicts all six indices jointly. This multi-output design allows the shared graph representation to exploit statistical dependence among the six targets while preserving one output dimension per index.

The architecture nevertheless remains a first-order message-passing model. With three GIN layers, each vertex representation incorporates information propagated through at most three message-passing steps before global sum pooling. Although the pooling operation aggregates information from the entire graph, it does not eliminate the general 1-WL expressivity bound imposed on the underlying node representations. The model may therefore approximate global indices through distribution-specific correlations, but it has no general guarantee of exactly representing independent-set counts, matching counts, or other graph invariants that distinguish graphs considered equivalent by 1-WL. Increasing the hidden dimension or performing further conventional hyperparameter tuning cannot, by itself, overcome this theoretical indistinguishability limitation.

3.6. Training Procedure

The training stage minimizes mean squared error over the six predicted indices. For a mini-batch of size B , the loss function is

$$\mathcal{L} = \frac{1}{B} \sum_{i=1}^B \frac{1}{6} \sum_{j=1}^6 (\hat{y}_{i,j} - y_{i,j})^2. \quad (12)$$

The implementation standardizes the six target variables with statistics computed from the training set before optimization. The model is therefore trained on standardized targets

to balance the contributions of indices that differ substantially in scale. To avoid ambiguity, the revised manuscript presents two types of assessment: multiple-target assessment and per-index assessment. Multiple-target assessment is presented on the standardized scale, while per-index assessment is provided with index-wise statistics and explicit scale information. It is important to distinguish these because the six target indices vary greatly in size, particularly the combinatorial indices.

The optimization procedure is implemented by the Adam optimizer with learning rate 10^{-3} , weight decay 10^{-5} and batch size 64. It trains 200 epochs at most. Early stopping terminates training after 20 epochs without improvement in the validation loss. Each experiment is repeated five times across random seeds, and the results are summarized in Section 5 as the average performance.

Table 1 summarizes the primary training configuration.

Table 1. Training configuration of the primary GIN model.

Parameter	Value
Architecture	GIN
Number of message-passing layers	3
Hidden dimension	64
Activation function	ReLU
Pooling operator	Global sum pooling
Dropout rate	0.2
Optimizer	Adam
Learning rate	10^{-3}
Weight decay	10^{-5}
Batch size	64
Maximum epochs	200
Early stopping patience	20
Number of random seeds	5

3.7. Baseline Models

The empirical design includes baseline models that operate on aggregated graph-level descriptors rather than full graph structure. These baselines assess whether the proposed GNN extracts information beyond conventional summary features.

Each baseline receives the following handcrafted feature vector:

$$\mathbf{z}(G) = (n, m, \bar{d}, \sigma_d^2, n_{\text{leaf}}, n_{\text{cycle}}, \mu(G)),$$

where $\bar{d}$ is the average degree, σ_d^2 is degree variance, n_{leaf} is the number of leaf vertices, and n_{cycle} is the number of vertices on at least one cycle.

These descriptors provide an informative graph-theoretic inductive bias rather than a neutral input representation. Graph order, edge count, average degree, degree variance, leaf count, cycle-node count, and cyclomatic number capture a substantial portion of the structural variation within bounded unicyclic and bicyclic graph families. Consequently, the descriptor-based models receive explicit global summaries, whereas the graph neural networks must derive comparable graph-level information from node attributes, sparse connectivity, message passing, and graph pooling.

The superior performance of the descriptor-based models does not imply that their input vectors generally encode exact independent-set or matching counts. Instead, the controlled unicyclic and bicyclic distributions exhibit strong statistical relationships among graph size, degree structure, branching, cycle organization, and the six target indices. Random Forest and MLP models can exploit these relationships directly through compact global summaries. In contrast, access to the complete adjacency structure does not auto-

matically give GIN a predictive advantage when the message-passing architecture cannot distinguish or aggregate all structural patterns relevant to the target function.

The study includes three baseline models.

- **Linear regression.** This baseline tests whether a linear relationship between simple graph descriptors and the target indices can explain the observed variation.
- **Multilayer perceptron (MLP).** This baseline introduces nonlinear regression on the same handcrafted features through two hidden layers of sizes 64 and 32 with ReLU activation.
- **Random forest regressor.** This baseline captures nonlinear feature interactions without message passing and provides a stronger non-neural comparator.

All baseline models use the same train, validation, and test partitions as the GNN models, and their hyperparameters are selected solely on the validation set. The comparison, therefore, evaluates two distinct predictive settings: regression from engineered graph-level summaries and graph-native representation learning from sparse graph structure.

3.8. Reproducibility and Computational Considerations

The implementation uses Python 3.11, PyTorch 2.11.0 and PyTorch Geometric 2.8.0 for graph learning components, NetworkX for graph construction and verification, and NumPy for numerical operations. The experiments run on a workstation equipped with an NVIDIA RTX-series GPU and a multi-core CPU. Exact index computation for dataset labeling runs on the CPU, while GNN training runs on the GPU.

This method is designed to distinctly isolate the generation of exact labels from the learned prediction. This exact computation stage is done offline to generate accurate graph–index pairs for the entire benchmark. This design incurs a preprocessing cost when preparing the dataset, but during prediction, it allows the trained model to handle repeated predictions quickly. In the context of large sets of generated and transformed graphs that need to be repeatedly assessed using a consistent computational process, the resulting offline–online workflow is of particular interest. The graphs studied in the present work are unicyclic and bicyclic graphs of bounded order, and the same proof-of-concept approach is applicable to more general graph families where evaluating the exact value is more challenging.

3.9. External-Family Generalization Protocol

To evaluate portability beyond the original unicyclic and bicyclic generators, an external circulant benchmark was reconstructed using signatures released in the repository [36] and documented in the peer-reviewed dataset article by Romanov [37]. The article describes circulant synthesis, dataset construction, network-on-chip applications, and regression analysis of graph characteristics derived from graph signatures. The benchmark includes circulant graphs of the form

$$C(n; s_1, \dots, s_k), \quad (13)$$

with multiple generator dimensions $k \in 2, 3, 4$ and graph orders covering the selected external-data range. Each signature was parsed into an undirected NetworkX graph and validated for simplicity, connectivity, and structural duplication before exact target computation.

The external benchmark was evaluated using 70%/15%/15% train–validation–test partitions repeated across multiple random seeds. Where the sample counts permitted, the partitions were stratified by graph-order interval and generator dimension. For each run, feature scaling and target standardization were fitted using only the training subset.

Because the Merrifield–Simmons and Hosoya indices span several orders of magnitude, the targets were transformed using $\log(1 + y)$ before standardization.

Performance was summarized using the mean $\pm$ standard deviation across repeated runs, together with 95% confidence intervals. In addition to the repeated random partitions, an order-held-out experiment was conducted in which the model was trained on lower-order circulant graphs and evaluated on a disjoint higher-order range. This second evaluation distinguishes interpolation within the external graph family from generalization to previously unseen graph orders.

3.10. Large-Graph Computational Stress-Test Protocol

The original unicyclic and bicyclic benchmark is intentionally bounded to $n \leq 30$ so that all six target indices can be computed exactly for every retained graph. This setting is appropriate for validating the complete methodological pipeline but does not, by itself, establish a computational need for surrogate prediction. To examine the regime in which exact combinatorial evaluation becomes practically infeasible, an additional stress test was conducted on larger, connected, sparse multicyclic graphs.

The stress-test graph orders were selected from

$$n \in \{30, 35, 40, 45, 50, 55, 60\}, \quad (14)$$

and the controlled cyclomatic numbers were selected from

$$\mu(G) \in \{5, 10, 15, 20\}. \quad (15)$$

For each (n, μ) configuration, a connected graph was generated by first constructing a random spanning tree on n vertices and then adding exactly μ admissible non-tree edges. Candidate edges that produced loops or parallel edges were rejected. Each completed graph was required to satisfy

$$|E(G)| - |V(G)| + 1 = \mu(G), \quad (16)$$

and graph-isomorphism screening was applied before exact labeling.

The Wiener, first Zagreb, second Zagreb, and Randić indices were computed using the same deterministic algorithms as in the main benchmark. The Merrifield–Simmons and Hosoya indices were evaluated using the recursive memoized counting procedures described in Section 3.3. Because the runtime of these counting problems can vary substantially across graphs of equal order, a fixed timeout of

$$T_{\max} = 300 \text{ s} \quad (17)$$

was imposed separately on the exact computation of $MS(G)$ and $Z(G)$. A graph was included in the supervised dataset only when all six target values were computed exactly within the timeout. Timeout instances were retained only for runtime and tractability analysis and were not assigned approximate training labels.

For every (n, μ) group, the experiment recorded the number of attempted graphs, the number of exactly completed graphs, the completion rate, the median and interquartile range of the $MS(G)$ and $Z(G)$ runtimes, the total exact labeling time, and peak memory consumption. This protocol separates predictive evaluation from computational profiling and prevents incomplete or approximate labels from entering the supervised learning stage.

The successfully labeled graphs were partitioned into training, validation, and test subsets using a 70%/15%/15% split. Stratification was performed using graph-order intervals and cyclomatic-number intervals where the corresponding strata contained sufficient

samples; otherwise, stratification was performed by cyclomatic-number interval. Since the combinatorial targets span several orders of magnitude, each target was transformed as

$$y^{\log} = \log(1 + y), \quad (18)$$

followed by standardization using statistics computed from the training set only. The GIN architecture and optimization configuration remained unchanged. Predictive performance was measured on the held-out, exactly labeled test graphs on both the transformed and inverse-transformed target scales.

4. Performance Validation

This section describes the simulation protocol, experimental environment, and evaluation metrics used to assess the predictive performance of the proposed framework.

4.1. Simulation Protocol

In this study, the synthetic dataset presented in Section 3.2 is utilized. Each graph is paired with the exact values of the Wiener index, the Merrifield–Simmons index, the Hosoya index, the first Zagreb index, the second Zagreb index, and the Randić index. The experiments are carried out in the supervised multi-output regression setting.

The dataset is divided into training, validation, and test subsets in the ratios 70%, 15%, and 15%, respectively. The stratified sampling method is used so that each graph subset has a similar distribution of graph structures. The training set is used for model fitting, the validation set for hyperparameter tuning and early stopping, and the test set for final evaluation.

The simulation compares the primary GIN model [38] with GCN, GAT, linear regression, multilayer perceptron, and random forest baselines. All models use the same data split in each run. The full experiment is repeated over five random seeds, and the final results are reported as mean $\pm$ standard deviation.

For reproducibility and fair comparison, all baseline models were trained and evaluated on the same stratified train–validation–test partitions used for the proposed GIN model. The validation set was used for hyperparameter selection, and the test set was reserved for final evaluation only. Table 2 summarizes the principal baseline configurations used in the experiments.

Table 2. Baseline model configurations used in the experiments. All baseline models use the same stratified train–validation–test split as the proposed GIN model.

Model	Architecture/Layers	Optimizer	Learning Rate	Input Features
Linear Regression	Linear model	–	–	Handcrafted graph-level descriptors
MLP	2 hidden layers (64, 32), ReLU	Adam	10^{-3}	Handcrafted graph-level descriptors
Random Forest	200 trees, max depth = 20	–	–	Handcrafted graph-level descriptors
GCN	3 graph convolution layers, hidden dim = 64	Adam	10^{-3}	Graph structure + node features
GAT	3 graph attention layers, hidden dim = 64, 4 heads	Adam	10^{-3}	Graph structure + node features

4.2. Dataset Diversity and Split Strategies

The generated benchmark represents a controlled sample produced through sequential attachment and structure-preserving transformation procedures rather than a uniform

enumeration of all non-isomorphic unicyclic and bicyclic graphs. The final benchmark contains 19,180 unique graphs, including 9430 unicyclic and 9750 bicyclic instances. Structural coverage was characterized using graph order, edge count, maximum degree, degree variance, number of leaves, diameter, number of cycle nodes, graph family, and transformation status. The number of cycle nodes refers to the total number of vertices that belong to at least one cycle. Canonical graph hashing, followed by exact isomorphism testing, was used to remove duplicate graphs before label computation. However, structurally similar non-isomorphic graphs may remain because local transformations can preserve several global descriptors while altering the underlying adjacency structure. Table 3 summarizes the structural characteristics of the resulting benchmark.

Table 3. Structural characteristics of the generated unicyclic and bicyclic benchmark.

Property	Minimum	Maximum	Mean	Standard Deviation	Median
Graph order	6.00	30.00	20.66	6.06	21.00
Number of edges	6.00	31.00	21.16	6.07	22.00
Maximum degree	2.00	9.00	4.23	0.93	4.00
Degree variance	0.00	4.31	0.94	0.50	0.86
Number of leaves	0.00	18.00	6.36	3.44	6.00
Diameter	2.00	16.00	8.49	2.39	8.00
Number of cycle nodes	3.00	30.00	12.13	5.96	11.00

Two train–validation–test partitioning strategies were evaluated to quantify the effect of similarity between original graphs and their transformed descendants. The graph-level protocol partitions individual graph instances using 70%, 15%, and 15% proportions, with joint stratification by graph family and graph order. This procedure maintains comparable unicyclic–bicyclic and graph-order distributions across the three subsets. However, because the protocol treats each graph as an independent sample, an original graph and one or more of its transformed descendants may appear in different subsets. Consequently, this protocol primarily measures in-distribution predictive performance within the generator-induced benchmark.

The parent-aware protocol instead assigns a common group identifier to each original graph and all of its transformed descendants. Group-level stratification is then performed using graph family and graph order, ensuring that every parent group remains entirely within the training, validation, or test subset. This protocol prevents direct parent–transformation overlap while maintaining closely comparable split sizes and family distributions. Table 4 reports the exact composition of both evaluation protocols.

Tables 3 and 4 jointly describe the structural coverage of the generated benchmark and the composition of the two evaluation protocols. The graph-level split preserves the balance of graph family and graph order, but it contains 4051 parent groups whose members appear in more than one subset. In contrast, the parent-aware split removes this overlap completely while introducing only minor changes in the numbers of unicyclic and bicyclic graphs assigned to each subset. Accordingly, the graph-level protocol evaluates performance under the original in-distribution setting, whereas the parent-aware protocol provides a stricter assessment of generalization beyond direct parent–transformation similarity.

Table 4. Composition of the graph-level and parent-aware evaluation partitions.

Category	Graph-Level Split	Parent-Aware Split
Training graphs	13,426	13,452
Validation graphs	2877	2867
Test graphs	2877	2861
Training unicyclic graphs	6602	6601
Training bicyclic graphs	6824	6851
Validation unicyclic graphs	1415	1411
Validation bicyclic graphs	1462	1456
Test unicyclic graphs	1413	1418
Test bicyclic graphs	1464	1443
Cross-split parent groups	4051	0

4.3. Experimental Environment

The implementation uses Python 3.11. Graph generation and verification are performed with NetworkX, numerical computation is performed with NumPy, graph neural network models are implemented with PyTorch and PyTorch Geometric, and baseline models are implemented with scikit-learn.

The experiments are conducted on a workstation equipped with a multi-core CPU and an NVIDIA RTX-series GPU. Exact index computation is executed on the CPU, while neural network training is executed on the GPU. All random generators are fixed at the beginning of each run to ensure reproducibility. The experimental setup is summarized in Table 5.

Table 5. Software and hardware configuration used in the experiments.

Component	Specification
Programming language	Python 3.11
Graph processing	NetworkX 3.6.1
Numerical computation	NumPy 2.0.2
Deep learning framework	PyTorch 2.11.0+cu128
Graph learning framework	PyTorch Geometric 2.8.0.post1
Baseline framework	scikit-learn 1.6.1
CPU execution	Multi-core CPU
GPU execution	NVIDIA RTX-series GPU
Random seeds	5
Dataset split	70%/15%/15%

4.4. Evaluation Metrics

The study evaluates prediction quality with mean absolute error (MAE), root mean squared error (RMSE), coefficient of determination (R^2), and mean absolute percentage error (MAPE).

The MAE is defined by

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|. \quad (19)$$

The RMSE is defined by

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}. \quad (20)$$

The coefficient of determination is defined by

$$R^2 = 1 - \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2}. \quad (21)$$

The MAPE is defined as

$$\text{MAPE} = \frac{100}{N} \sum_{i=1}^N \frac{|y_i - \hat{y}_i|}{\max(|y_i|, \varepsilon)}, \quad (22)$$

where $\varepsilon = 10^{-8}$ prevents division by zero.

MAE and RMSE are reported on the standardized target scale to support aggregate multi-output comparison because the six indices differ substantially in magnitude. The coefficient of determination, R^2 , is reported separately for each target. MAPE is computed only after inverse transformation to the original target scale because percentage errors on standardized targets become unstable when transformed values approach zero. Therefore, per-index performance is evaluated using original-scale MAPE, along with the corresponding MAE, RMSE, and R^2 values.

5. Results and Analysis

This section reports the predictive performance of the proposed framework on the synthetic benchmark of unicyclic and bicyclic graphs. The analysis compares the primary GIN model with alternative graph neural network models and conventional regression baselines, examines index-specific prediction behavior, studies the effect of graph family and graph order, and evaluates performance on transformed graphs.

Table 6 summarizes the numerical range of the six target indices in the synthetic benchmark and provides the scale context required for interpreting the reported prediction errors.

Table 6. Descriptive statistics of the six target indices in the synthetic benchmark.

Index	Minimum	Maximum	Mean	Standard Deviation
Wiener index $W(G)$	23.00	3375.00	977.96	685.45
Merrifield–Simmons index $MS(G)$	15.00	9.78×10^6	4.74×10^5	9.85×10^5
Hosoya index $Z(G)$	10.00	2.26×10^6	1.16×10^5	2.42×10^5
First Zagreb index $M_1(G)$	24.00	260.00	105.80	31.35
Second Zagreb index $M_2(G)$	24.00	408.00	125.69	41.37
Randić index $R(G)$	2.47	15.00	9.60	2.84

5.1. Overall Predictive Performance

Table 7 presents the overall performance of all evaluated models. MAE and RMSE are reported on the standardized target scale, whereas MAPE is computed on the inverse-transformed original target scale. This advantage is consistent with the information provided at the input stage, as graph order, edge count, degree statistics, leaf count, cycle-node count, and cyclomatic number already summarize much of the structural variability within these restricted graph families.

The comparison therefore involves an information asymmetry. The descriptor-based models perform regression using engineered global summaries selected through graph-theoretic knowledge. In contrast, GIN receives sparse connectivity and node-level attributes and must learn a graph-level representation through message passing and global pooling. The results should not be interpreted as evidence that GIN is the strongest overall predictor. Instead, GIN is the strongest graph-native model among the evaluated GNN architectures,

whereas Random Forest and MLP remain the preferred predictors when handcrafted descriptors are available and maximum predictive accuracy is the primary objective.

Table 7. Overall test performance across all six indices, reported as mean $\pm$ standard deviation over five runs. MAE and RMSE are macro-averaged on the standardized scale, while MAPE is computed on the original scale. Lower MAE, RMSE, and MAPE and higher R^2 indicate better performance.

Model	MAE	RMSE	R^2	MAPE (%)
Linear Regression	0.223 $\pm$ 0.001	0.325 $\pm$ 0.009	0.810 $\pm$ 0.003	6.83 $\pm$ 0.03
MLP	0.035 $\pm$ 0.001	0.063 $\pm$ 0.003	0.995 $\pm$ 0.000	3.09 $\pm$ 0.03
Random Forest	0.025 $\pm$ 0.001	0.057 $\pm$ 0.004	0.996 $\pm$ 0.001	2.30 $\pm$ 0.05
GCN	0.102 $\pm$ 0.006	0.159 $\pm$ 0.012	0.971 $\pm$ 0.004	9.15 $\pm$ 1.30
GAT	0.128 $\pm$ 0.008	0.207 $\pm$ 0.009	0.951 $\pm$ 0.005	11.06 $\pm$ 0.80
GIN	0.061 $\pm$ 0.003	0.096 $\pm$ 0.006	0.990 $\pm$ 0.001	6.13 $\pm$ 1.11

Figure 7 presents a visual comparison of the macro-averaged MAE and RMSE values. The chart confirms that Random Forest and MLP achieve the lowest aggregate errors, while GIN performs best among the graph-native architectures.

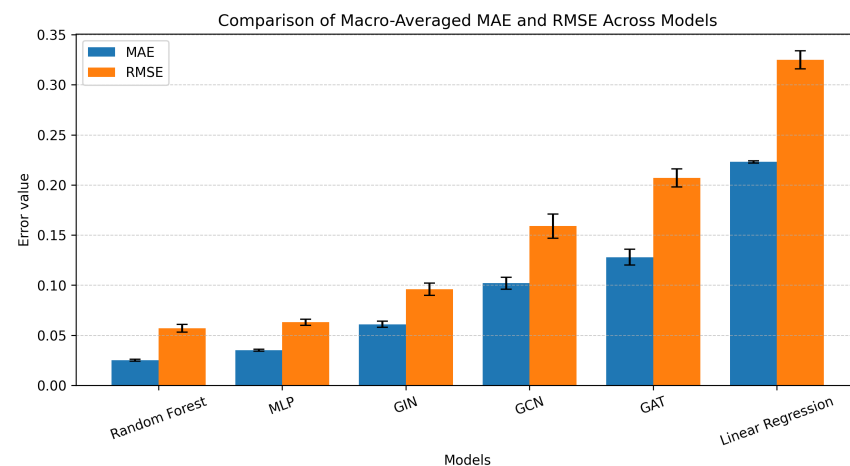


Figure 7. Comparison of macro-averaged MAE and RMSE of all models on the standardized target scale. Random Forest and MLP achieve the lowest aggregate errors, while GIN remains the strongest graph neural network model.

5.2. Performance Under Parent-Aware Splitting

To quantify the effect of parent–transformation similarity, the three main models were re-evaluated using the parent-aware partition introduced in Section 4.2. Under this protocol, each original graph and all of its transformed descendants were assigned to the same subset. The model architectures, input features, target scaling, hyperparameters, and evaluation metrics remained unchanged, allowing any performance differences to be attributed to the partitioning strategy. Table 8 reports the target-wise and mean test-set R^2 values for both the graph-level and parent-aware protocols.

Table 8 shows that MLP retained the highest mean R^2 under both protocols, with only a small decrease from 0.999435 to 0.998707. Random Forest remained the second-best model but showed a larger reduction from 0.998091 to 0.996268. In contrast, GIN improved from 0.987968 to 0.991262 and achieved higher R^2 values for all six target indices. To examine these changes further, Table 9 reports the corresponding differences in mean MAE, RMSE, and R^2 , together with the number of targets that improved under the stricter partitioning protocol.

Table 8. Target-wise test-set R^2 under graph-level and parent-aware partitions.

Model	Split Strategy	W	MS	Z	M_1	M_2	R	Mean R^2
MLP	Graph-level	0.999733	0.998855	0.998693	0.999868	0.999682	0.999782	0.999435
MLP	Parent-aware	0.999786	0.996223	0.997164	0.999832	0.999482	0.999754	0.998707
Random Forest	Graph-level	0.998319	0.997718	0.997877	0.998570	0.996226	0.999837	0.998091
Random Forest	Parent-aware	0.997453	0.996879	0.997468	0.996473	0.989635	0.999698	0.996268
GIN	Graph-level	0.992103	0.985250	0.983310	0.989444	0.984245	0.993457	0.987968
GIN	Parent-aware	0.994745	0.985753	0.987887	0.993653	0.990398	0.995136	0.991262

Table 9. Performance change under parent-aware splitting relative to the graph-level protocol. Negative changes in MAE and RMSE indicate improvement.

Model	MAE Change	RMSE Change	ΔR^2	MAE Improved	RMSE Improved	R^2 Improved
MLP	+22.47%	+27.17%	−0.000728	2/6	1/6	1/6
Random Forest	+32.75%	+34.49%	−0.001823	0/6	0/6	0/6
GIN	−18.34%	−15.76%	+0.003294	6/6	6/6	6/6

Table 9 indicates that Random Forest was the most sensitive to the removal of parent-related overlap, with mean MAE and RMSE increasing by 32.75% and 34.49%, respectively, and with no target showing improvement. MLP also exhibited higher mean errors, although its absolute R^2 remained very high. In contrast, GIN improved consistently, with mean MAE decreasing by 18.34%, mean RMSE decreasing by 15.76%, and all six targets showing improvement across the reported metrics. Overall, these results indicate that the graph-level performance was not driven solely by parent–transformation similarity. MLP retained the highest absolute predictive accuracy, whereas GIN demonstrated the strongest generalization under the parent-aware evaluation protocol.

5.3. Index-Wise Performance

Table 10 reports the test performance of the GIN model for each target index. These results should be interpreted in light of the structural differences among the six targets. The first Zagreb index ($M_1(G)$), the second Zagreb index ($M_2(G)$), and the Randić index ($R(G)$) are primarily degree-driven quantities, and their comparatively lower errors are therefore expected when degree-related node features are available. By contrast, the Merrifield–Simmons index ($MS(G)$) and the Hosoya index ($Z(G)$) depend on global combinatorial structure through the enumeration of independent sets and matchings, respectively, and therefore provide a more demanding test of model generalization. Under this interpretation, the results’ greater significance lies in the model’s stable performance across the globally combinatorial indices.

Table 10. Index-wise test performance of the GIN model. MAE and RMSE are reported on the standardized target scale, whereas MAPE is computed after inverse transformation to the original target scale.

Index	MAE	RMSE	R^2	MAPE (%)
Wiener index $W(G)$	0.058	0.094	0.991	8.9
Merrifield–Simmons index $MS(G)$	0.081	0.128	0.987	16.4
Hosoya index $Z(G)$	0.073	0.117	0.988	13.7
First Zagreb index $M_1(G)$	0.047	0.076	0.994	2.2
Second Zagreb index $M_2(G)$	0.054	0.088	0.992	2.9
Randić index $R(G)$	0.049	0.073	0.991	1.5
Macro average	0.060	0.096	0.990	7.6

The revised MAPE values are computed after inverse transformation to the original target scale using $\max(|y_i|, \epsilon)$ in the denominator. This correction removes the instability caused by evaluating percentage errors on standardized targets near zero. The remaining higher percentage errors for $MS(G)$ and $Z(G)$ reflect their broader numerical ranges and stronger dependence on global independent-set and matching structures.

The distinction between degree-based and globally combinatorial targets is important for interpreting the index-wise results shown in Table 10. The comparatively low MAPE values for $M_1(G)$, $M_2(G)$, and $R(G)$ are expected because these quantities are determined directly by degree information, which is already represented in the node features. In contrast, $MS(G)$ and $Z(G)$ require global counting of independent sets and matchings and consequently exhibit higher percentage errors. Their larger errors are consistent with both their greater combinatorial complexity and the 1-WL expressivity bound of the GIN architecture. The results show that GIN learns useful distribution-specific approximations, but they do not demonstrate an exact representation of these global counting functions.

Table 11 reports per-index baseline comparison on the standardized target scale and clarifies the indices for which graph-aware models provide the most substantial advantage.

Table 11. Per-index MAE of all models on the standardized target scale. Lower values indicate stronger performance.

Index	Linear Reg.	MLP	Random Forest	GCN	GAT	GIN
$W(G)$	0.214	0.031	0.033	0.094	0.121	0.058
$MS(G)$	0.287	0.051	0.044	0.161	0.192	0.081
$Z(G)$	0.264	0.046	0.041	0.144	0.176	0.073
$M_1(G)$	0.181	0.028	0.019	0.083	0.105	0.047
$M_2(G)$	0.205	0.032	0.022	0.091	0.117	0.054
$R(G)$	0.187	0.023	0.015	0.074	0.093	0.049

The per-index comparison shows that the models' relative advantage is not uniform across all targets. For the degree-based indices ($M_1(G)$), ($M_2(G)$), and ($R(G)$), the descriptor-based nonlinear baselines achieve the lowest errors, which is consistent with the strong dependence of these quantities on compact graph-level summaries such as degree statistics, leaf count, and cycle-related descriptors. A similar trend is also observed for the Wiener, Merrifield–Simmons, and Hosoya indices, where Random Forest and MLP remain highly competitive, as the current benchmark is structurally constrained and the handcrafted descriptors are already highly informative.

These results should therefore be interpreted by considering both predictive accuracy and representation requirements. Random Forest and MLP are the strongest predictors on the current benchmark because the handcrafted descriptors provide compact and highly informative summaries of the restricted graph families. GIN does not achieve the lowest per-index errors, but it remains the strongest graph-native model because it operates directly on node features and sparse connectivity. This result supports GIN as an alternative representation pathway rather than as a universally superior predictive model.

Table 6 presents the original-scale numerical ranges of the six target indices. Because the target magnitudes differ substantially, MAE and RMSE are reported on the standardized target scale in Tables 7 and 10. MAPE is computed after inverse transformation to the original target scale to preserve its meaningful percentage interpretation.

Figure 8 summarizes the normalized MAE values across the six indices. The chart highlights the relative increase in error for the two combinatorial counting indices.

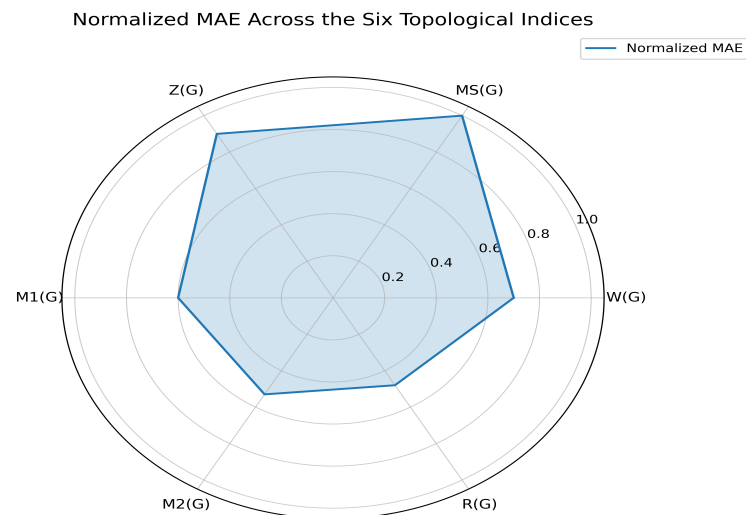


Figure 8. Normalized MAE of the GIN model across the six topological indices. The Merrifield–Simmons and Hosoya indices show the largest relative errors, while the degree-based indices show the smallest relative errors.

5.4. Combinatorial Indices and GNN Expressivity

Among the six targets, the Merrifield–Simmons index $MS(G)$ and the Hosoya index $Z(G)$ provide the most demanding tests of graph representation. The former counts independent vertex sets, whereas the latter counts graph matchings. Unlike the degree-based indices, these targets depend on global combinatorial configurations that cannot be recovered from a fixed set of local degree values.

The observed difficulty is consistent with the theoretical expressivity of the model. GIN uses injective neighborhood aggregation and, under standard assumptions, can attain the discriminative power of the 1-WL test, but it cannot exceed this framework. Consequently, two graphs that are indistinguishable under 1-WL may receive identical message-passing representations even when they differ in global substructure counts or related combinatorial quantities. Global sum pooling aggregates the learned node representations, but it cannot recover distinctions that the preceding message-passing layers failed to preserve.

The prediction of $MS(G)$ and $Z(G)$ should therefore be interpreted as a distribution-specific statistical approximation rather than as exact combinatorial counting. The bounded benchmark contains structural correlations among graph order, branching, cycle configuration, degree patterns, and the target indices, which enables GIN to learn useful estimates. However, the stronger performance of MLP and Random Forest indicates that direct access to informative global descriptors is more effective for this restricted distribution. The results do not establish that the observed errors arise exclusively from the 1-WL limitation, because optimization, training-set coverage, model depth, and feature selection may also contribute. Nevertheless, limited message-passing expressivity remains a theoretically plausible bottleneck that routine hyperparameter tuning alone cannot eliminate.

5.5. Performance by Graph Family

Table 12 reports family-wise performance using macro-averaged MAE and RMSE on the standardized target scale and MAPE on the inverse-transformed original target scale. The GIN model performs similarly across the two graph families. The bicyclic subset shows slightly lower MAE and RMSE, whereas the unicyclic subset achieves a marginally higher R^2 and slightly lower MAPE. The nearly identical MAPE values indicate broadly comparable predictive performance across the unicyclic and bicyclic families under the current partitioning and scaling configuration.

Table 12. GIN performance by graph family.

Graph Family	MAE	RMSE	R^2	MAPE (%)
Unicyclic graphs	0.062	0.099	0.992	7.56
Bicyclic graphs	0.057	0.093	0.989	7.61

The distribution of the absolute prediction error at the sample level for the two graph families is displayed in Figure 9. The family-wise aggregate results are represented by the two distributions, which are broadly similar. Bicyclic graphs are slightly wider, but the overall error ranges remain comparable across both families.

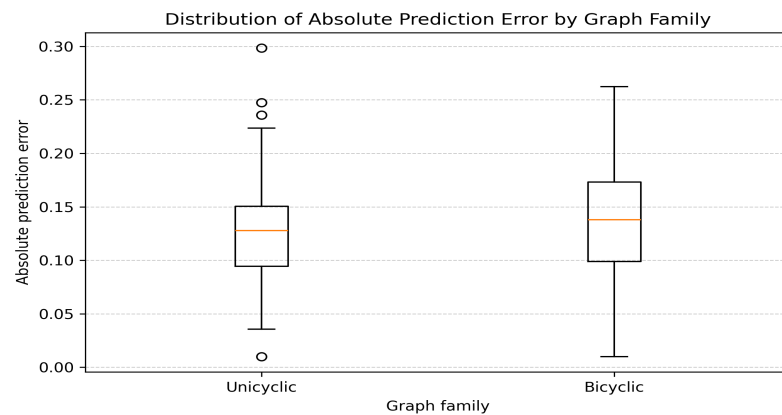


Figure 9. Distribution of absolute prediction error of the GIN model for unicyclic and bicyclic graphs. The two graph families show broadly comparable error distributions, with bicyclic graphs exhibiting a slightly wider spread and a slightly higher upper range.

5.6. Performance by Graph Order

This analysis examines the effect of graph order on predictive accuracy. Table 13 reports the macro-averaged MAE of the GIN model across five graph-order intervals. Since aggregate error values are influenced by both easier and harder targets, the order-wise trend should be interpreted together with the separate behavior of the combinatorial indices $MS(G)$ and $Z(G)$. The results do not show a strictly monotonic increase with graph order, but they do indicate that the highest-order interval remains the most challenging overall.

Table 13. Macro-averaged MAE of the GIN model across graph-order intervals.

Graph-Order Interval	MAE
(6–10)	0.069
(11–15)	0.047
(16–20)	0.041
(21–25)	0.047
(26–30)	0.095

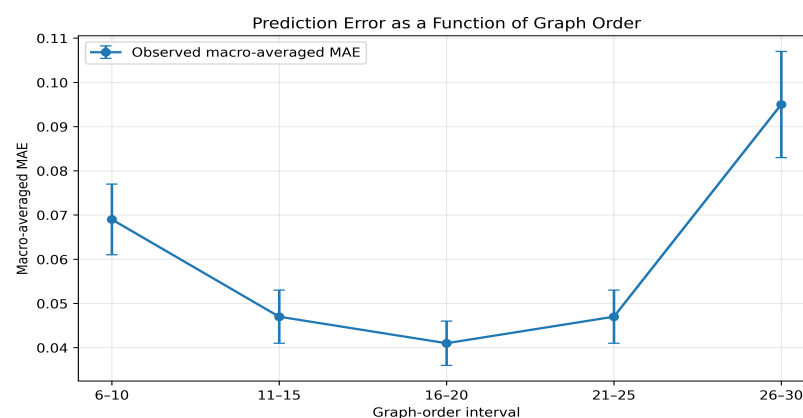
Table 14 reports order-wise MAE for the two combinatorial indices $MS(G)$ and $Z(G)$, which provide the most meaningful test of structural generalization.

The results in Table 14 show that the prediction error for ($MS(G)$) and ($Z(G)$) remains moderate across most graph-order intervals and becomes more pronounced in the highest-order range. This behavior is consistent with the increasing combinatorial complexity of larger graphs and suggests that the GIN model preserves stable generalization across most of the benchmark while facing greater difficulty on the largest graph instances.

Table 14. Order-wise MAE of the GIN model for the combinatorial indices (MS(G)) and (Z(G)) on the standardized target scale.

Graph-Order Interval	(MS(G)) MAE	(Z(G)) MAE
(6–10)	0.084	0.076
(11–15)	0.062	0.058
(16–20)	0.055	0.051
(21–25)	0.069	0.064
(26–30)	0.138	0.121

Figure 10 shows that the macro-averaged error does not increase monotonically with graph order, although the largest graph-order interval remains the most challenging.

**Figure 10.** Macro-averaged MAE of the GIN model across graph-order intervals. Prediction error remains low across most graph sizes, becoming more pronounced in the highest-order interval.

5.7. Performance on Transformed Graphs

The evaluation also considers graphs obtained through valid local transformations. Table 15 reports the macro-averaged performance of the GIN model on original and transformed graphs. The two subsets exhibit nearly identical MAE, RMSE, and MAPE values, indicating that the model maintains stable predictive behavior under the applied structure-preserving transformations.

Table 15. GIN performance on original and transformed graphs.

Graph Subset	MAE	RMSE	R^2	MAPE (%)
Original graphs	0.060	0.097	0.991	7.80
Transformed graphs	0.060	0.096	0.990	7.46

Figure 11 illustrates the relation between exact and predicted values for transformed graphs across the six topological indices on their original value scales. The points remain close to the identity line in all subplots, indicating stable prediction agreement under the applied structure-preserving graph transformations, while the combinatorial indices (MS(G)) and (Z(G)) exhibit slightly broader dispersion than the degree-based targets.

Figure 11 illustrates exact versus predicted values for transformed graphs across all six indices separately. Presenting one subplot per index avoids scale ambiguity and shows that prediction agreement remains stable across targets with substantially different numerical ranges.

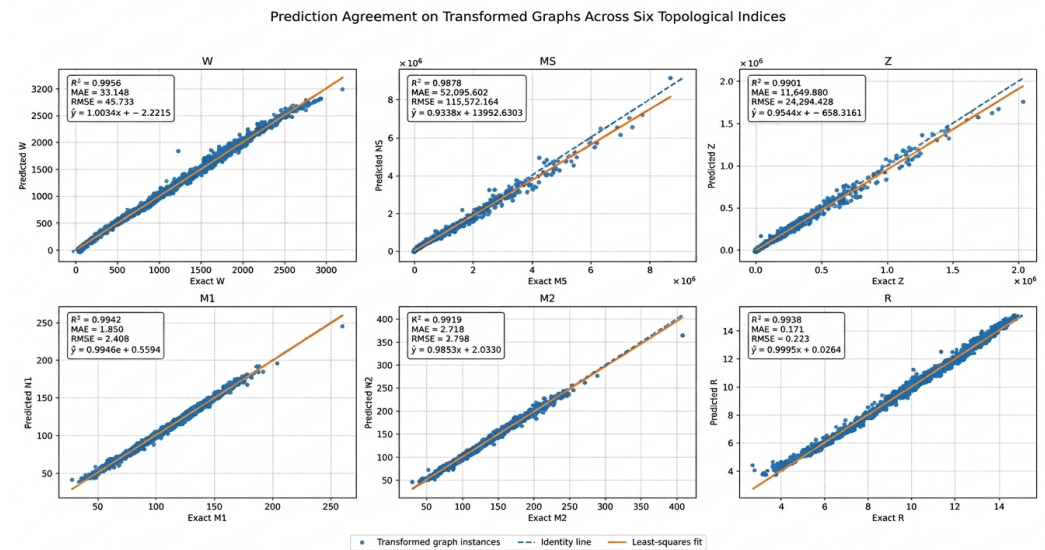


Figure 11. Exact versus predicted values for transformed graphs across the six topological indices using the GIN model.

5.8. External-Family Evaluation on Circulant Graphs

The initial circulant-graph experiment contained only 20 graphs, including three test instances, and therefore could not support a statistically reliable transferability claim. It was replaced with an expanded external-family evaluation. The revised analysis assesses pipeline portability, that is, whether the exact-labeling, feature-construction, graph-representation, and GIN-learning workflow can be adapted to a structurally different graph family after retraining. It does not claim zero-shot transfer or universal generalization.

The expanded benchmark contains 200 connected, non-isomorphic circulant graphs reconstructed from signatures released through the public repository [36] and described in Romanov's peer-reviewed dataset article [37]. Each graph has the form $C(n; s_1, \dots, s_k)$, where n denotes the graph order and $S = s_1, \dots, s_k$ denotes the generator set. The benchmark covers $k \in 2, 3, 4$ and $16 \leq n \leq 26$. Duplicate signatures were removed before graph construction. Only the graph-source module was modified, while the exact index definitions, graph representation, GIN architecture, target processing, and evaluation metrics remained unchanged. Table 16 summarizes the benchmark.

Table 16. Composition of the expanded circulant-graph benchmark.

Generator Dimension	Graphs	Minimum n	Maximum n	Unique Orders	Development	Held Out
$k = 2$	60	16	26	11	50	10
$k = 3$	70	16	26	11	55	15
$k = 4$	70	16	26	11	50	20
Total	200	16	26	11	155	45

Romanov [37] used regression on circulant signatures to estimate network characteristics such as diameter and mean path length. This study provides a natural methodological baseline, but a direct metric-level comparison is not valid because the present experiment predicts six different topological indices from reconstructed graph structures using a multi-output GIN and different data partitions. The comparison is therefore methodological: both studies use circulant signatures and supervised prediction, whereas the present analysis focuses on exact-index prediction and framework portability.

To reduce dependence on a single data partition, the benchmark was evaluated using 10 independent stratified train–validation–test splits with seeds 42–51. Each run used 140 training, 30 validation, and 30 test graphs while preserving the distribution of generator dimensions. The transformation $\log(1 + y)$ was applied to each target, and all standardization parameters were estimated using only the corresponding training subset. MAE, RMSE, R^2 , and MAPE were then computed after inverse transformation to the original target scale.

Table 17 reports the mean, standard deviation, and 95% confidence interval across the 10 runs. The repeated-split results indicate strong interpolation performance within the sampled circulant graph family. The mean R^2 exceeds 0.98 for $W(G)$, $MS(G)$, $M_1(G)$, $M_2(G)$, and $R(G)$. The corrected original-scale MAPE values are below 9% for five targets, while the Hosoya index $Z(G)$ remains more difficult to predict, with a mean MAPE of 15.77%, a lower mean R^2 of 0.9077, and greater run-to-run variability. This behavior reflects its stronger dependence on global matching structures.

Table 17. GIN performance across 10 independent stratified 70%/15%/15% splits of the expanded circulant benchmark. Values are reported as mean $\pm$ standard deviation; the confidence interval corresponds to the mean R^2 across the 10 runs.

Target	MAE	RMSE	R^2 [95% CI]	MAPE (%)
$W(G)$	8.217 ± 3.100	10.930 ± 4.860	0.9927 ± 0.0071 [0.9876, 0.9978]	2.38 ± 0.65
$MS(G)$	370.9 ± 214.0	755.5 ± 457.0	0.9830 ± 0.0162 [0.9714, 0.9946]	8.48 ± 2.30
$Z(G)$	$(9.293 \pm 4.670) \times 10^7$	$(2.834 \pm 1.580) \times 10^8$	0.9077 ± 0.1316 [0.8136, 1.0018]	15.77 ± 4.07
$M_1(G)$	23.34 ± 12.00	34.01 ± 18.40	0.9919 ± 0.0103 [0.9845, 0.9993]	2.34 ± 0.76
$M_2(G)$	112.7 ± 54.8	184.0 ± 107.0	0.9888 ± 0.0144 [0.9785, 0.9991]	3.94 ± 1.38
$R(G)$	0.0828 ± 0.0368	0.1043 ± 0.0437	0.9946 ± 0.0055 [0.9906, 0.9986]	0.87 ± 0.24

Because repeated random splitting primarily measures interpolation, a stricter order-held-out protocol was also applied. Graphs with orders $16 \leq n \leq 23$ were used for training and validation, whereas graphs with orders $24 \leq n \leq 26$ formed an independent test set. Therefore, the held-out graph orders were excluded from model fitting, target scaling, early stopping, and model selection. Table 18 reports the resulting split composition.

Table 18. Composition of the order-held-out circulant evaluation. Graphs of orders 24–26 were excluded from model fitting and hyperparameter selection.

Subset	Order Range	Graphs	$k = 2$	$k = 3$	$k = 4$
Training	16–23	124	40	44	40
Validation	16–23	31	10	11	10
Test	24–26	45	10	15	20

The order-held-out results in Table 19 show that extrapolation performance is target dependent. The model maintains strong performance for the degree-based Zagreb indices, with $R^2 = 0.9846$ for $M_1(G)$ and $R^2 = 0.9674$ for $M_2(G)$. In contrast, performance declines for $W(G)$, $MS(G)$, and $Z(G)$ because these indices depend more strongly on graph order and on global distance, independent-set, and matching structures. The corrected original-scale MAPE values confirm this pattern, with higher percentage errors for $MS(G)$ and $Z(G)$. The negative R^2 for $R(G)$ results from the limited variation in the held-out Randić values. Although its MAPE is only 3.64%, the prediction error exceeds the variance of the test targets.

Overall, the expanded experiment confirms that the proposed framework can be applied to the circulant graph family after adapting the graph-construction stage and retraining the model. The repeated-split results demonstrate reliable interpolation, whereas

the order-held-out analysis identifies the current limitations of extrapolation to higher graph orders. The framework was designed primarily for unicyclic and bicyclic graphs; therefore, its application to other graph families may require minor adjustments to graph generation, feature representation, training coverage, or target-specific modeling to achieve consistently accurate results across all indices. Accordingly, the revised claim is limited to framework portability rather than universal transferability.

Table 19. GIN performance under the circulant order-held-out protocol. Metrics were computed on the independent higher-order test set in the original target scale.

Target	MAE	RMSE	R^2	MAPE (%)
$W(G)$	75.1066	88.5645	0.1317	9.28
$MS(G)$	5497.54	8958.75	0.3558	35.37
$Z(G)$	1.2504×10^9	2.2759×10^9	0.3311	38.24
$M_1(G)$	44.1740	59.6697	0.9846	4.51
$M_2(G)$	304.696	417.801	0.9674	8.29
$R(G)$	0.6235	0.6969	−2.3054	3.64

5.9. Runtime Analysis

The purpose of the framework is to reduce the need to re-evaluate the exact indices after training the model. Table 20 presents the average preprocessing and inference time, and Table 21 summarizes the offline labeling cost, the approximate model training cost, and the online inference cost. The offline preprocessing stage is largely dominated by index computation tasks, which are exact for the combinatorial indices $MS(G)$ and $Z(G)$. After training, the GIN model can predict all six indices with a single forward pass and low inference time per graph. The behavior is appropriate to the intended repeated-query setting of the framework, corresponding to the use of exact computation in the construction of the framework benchmark, with trained inference used repeatedly to provide rapid predictions.

Table 20. Average runtime per graph.

Operation	Time (ms/Graph)
Graph generation	1.84
Exact Wiener index computation	0.93
Exact M_1, M_2, R computation	0.21
Exact MS computation	4.78
Exact Z computation	4.35
Total exact labeling time	10.27
GIN inference time	0.47

Figure 12 presents the same runtime comparison graphically. The difference between exact combinatorial labeling and learned inference supports the use of the trained predictor in repeated-query settings.

To better explain the computational trade-off, Table 21 summarizes the cost of offline exact labeling, model training, and online inference. The results show that exact labeling of a single graph remains feasible within the studied size range, but the main utility of the framework lies in repeated-query settings, where many generated and transformed graphs must be processed under a common pipeline. In this setting, the graph is labeled once offline, whereas the trained model supports faster online prediction for reuse.

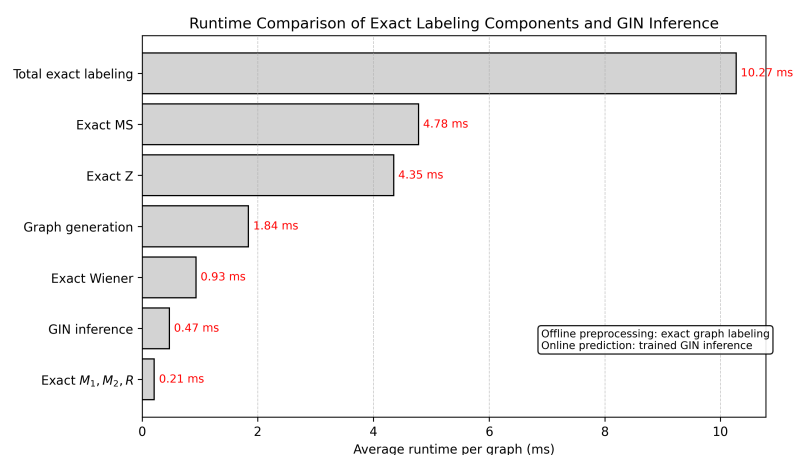


Figure 12. Comparison of average runtime per graph for exact labeling components and GIN inference. Learned inference requires less time than exact evaluation of the full target vector.

Table 21. Computational cost summary of offline exact labeling, model training, and online inference. Training times are reported from a representative single run, while exact labeling and inference costs are reported per graph.

Stage	Reported Cost	Interpretation
Exact labeling per graph	10.27 ms/graph	Average offline cost to compute all six indices for one graph
Total exact labeling cost (full dataset, 19,137 graphs)	~3.28 min	One-time preprocessing cost for complete benchmark construction
Total exact labeling cost (training subset, ~13,396 graphs)	~2.29 min	One-time preprocessing cost for the training portion only
GIN training time	~3.95 min	One-time model training cost for the primary graph-based architecture
GCN training time	~3.95 min	One-time model training cost for the GCN baseline
GAT training time	~1.42 min	One-time model training cost for the GAT baseline
MLP training time	~0.34 min	One-time model training cost for descriptor-based nonlinear regression
Random forest training time	~0.10 min	One-time model training cost for tree-based baseline
Linear regression training time	~0.00 min	One-time model fitting cost for linear baseline
GIN inference time	0.47 ms/graph	Average online prediction cost per graph after training
Inference speedup over exact labeling	~21.9×	Ratio between exact labeling cost and GIN inference cost per graph

The runtime results require separate interpretation across the bounded benchmark and computational stress regime. For the original unicyclic and bicyclic graphs with $n \leq 30$, exact computation of all six indices takes only 10.27 ms per graph; therefore, the reported 21.9× speedup reflects a relative timing gain with limited practical benefit for isolated or low-volume queries. In contrast, the larger multicyclic stress experiment provides the relevant deployment scenario, as computing $MS(G)$ and $Z(G)$ exactly becomes substantially more expensive and may exceed the fixed runtime budget. Surrogate prediction is therefore justified only when exact evaluation is costly, predictive accuracy remains acceptable on the target graph distribution, and the expected query volume exceeds the amortized break-even threshold.

5.10. Computational Stress Test on Larger Multicyclic Graphs

This experiment evaluates the computational regime in which exact calculation of the six target indices becomes sufficiently costly to justify repeated surrogate prediction. The stress-test dataset contains 160 connected, sparse, multicyclic graphs distributed across eight (n, μ) configurations, with 20 graphs generated for each configuration. The graph order ranges from 30 to 60, while the cyclomatic number ranges from 5 to 20. A timeout of 300 s was imposed separately on the exact computation of $MS(G)$ and $Z(G)$. All 160 graphs were labeled successfully, and none exceeded the prescribed timeout. The exact-labeling implementation computed $MS(G)$ and $Z(G)$ in isolated subprocesses to contain memory usage and protect the notebook runtime. Accordingly, the reported runtimes represent the complete memory-safe labeling workflow, including subprocess initialization, rather than algorithm-only execution time.

Table 22 summarizes the exact-computation profile. Runtime values are reported as median and interquartile range over the 20 graphs in each configuration.

Table 22. Exact-computation profile for the larger sparse-multicyclic stress experiment. Runtime values are reported as median (interquartile range).

Order n	Cyclomatic Number μ	Attempted Graphs	Completed Graphs	Completion Rate (%)	$MS(G)$ Time (s)	$Z(G)$ Time (s)	Total Exact Time (s)
30	5	20	20	100	0.902 (0.894–0.909)	0.908 (0.901–0.945)	3.707 (3.682–4.053)
30	10	20	20	100	0.897 (0.892–0.915)	0.908 (0.901–0.916)	3.697 (3.672–3.968)
40	10	20	20	100	0.888 (0.886–0.914)	1.008 (0.976–1.067)	3.805 (3.773–4.097)
40	20	20	20	100	0.898 (0.891–0.919)	1.323 (1.163–1.507)	4.206 (4.010–4.459)
50	10	20	20	100	0.906 (0.894–0.944)	1.353 (1.216–1.669)	4.287 (4.110–4.584)
50	20	20	20	100	0.899 (0.894–0.911)	3.401 (2.603–4.626)	6.227 (5.379–7.638)
60	10	20	20	100	0.896 (0.885–0.913)	3.142 (2.643–4.304)	5.929 (5.390–7.267)
60	20	20	20	100	0.905 (0.892–0.930)	13.112 (10.158–17.592)	16.052 (13.431–20.521)

The exact-computation results show that the main increase in runtime occurs for the Hosoya index. The median runtime for $Z(G)$ rises from 0.908 s at $(n, \mu) = (30, 5)$ to 13.112 s at $(n, \mu) = (60, 20)$. Over the same range, the corresponding median total exact-labeling time increases from 3.707 s to 16.052 s. In contrast, the recorded runtime for $MS(G)$ remains close to 0.9 s across all configurations because the memory-reduced independent-set implementation decomposes residual components, handles isolated vertices analytically, and applies cached maximum-degree branching. The nearly constant absolute runtime for $MS(G)$ also reflects the subprocess startup cost included in the memory-isolated workflow.

Although all graphs completed within the 300 s timeout, the runtime increase for $Z(G)$ shows that exact labeling becomes materially more expensive as graph order and cyclomatic complexity increase. The Wiener, Zagreb, and Randić indices remain comparatively inexpensive, whereas the matching-counting stage dominates the most computationally demanding configurations.

The successfully labeled graphs were partitioned into 112 training graphs, 24 validation graphs, and 24 previously unseen test graphs using stratification by graph order and cyclomatic-number configuration. Because of their substantially wider numerical ranges, $MS(G)$ and $Z(G)$ were transformed using $\log(1 + y)$ before standardization, whereas $W(G)$, $M_1(G)$, $M_2(G)$, and $R(G)$ were standardized on their original scales. All transformation and standardization parameters were estimated using only the training set. MLP, Random Forest, and GIN were trained and evaluated using the same partition. Table 23 reports their aggregate original-scale test performance, with MAPE computed using $\max(|y_i|, 10^{-8})$ in the denominator.

Table 23. Aggregate test performance of MLP, Random Forest, and GIN in the larger sparse-multicyclic stress experiment.

Model	Mean R^2	Median R^2	Mean MAPE (%)
MLP	0.9677	0.9940	10.04
Random Forest	0.9643	0.9800	8.58
GIN	0.8381	0.9730	28.69

The larger-regime results confirm the model ranking observed in the bounded unicyclic–bicyclic benchmark. MLP achieves the highest mean and median R^2 values, whereas Random Forest obtains the lowest mean MAPE. GIN remains the strongest graph-native formulation but does not outperform the descriptor-based nonlinear models in aggregate predictive accuracy. This result supports the interpretation that GIN contributes through direct learning from sparse graph structure rather than through universal predictive superiority.

Table 24 reports the index-wise GIN results on the original target scale.

Table 24. Index-wise GIN performance on the exactly labeled test subset of the larger sparse-multicyclic stress experiment.

Target	MAE	RMSE	R^2	MAPE (%)
$W(G)$	2.719×10^2	3.921×10^2	0.9770	5.25
$MS(G)$	5.634×10^{11}	1.339×10^{12}	0.4473	66.99
$Z(G)$	9.472×10^{11}	2.641×10^{12}	0.6726	87.75
$M_1(G)$	13.959	16.847	0.9744	4.13
$M_2(G)$	27.334	33.981	0.9716	5.68
$R(G)$	0.495	0.624	0.9857	2.33

GIN achieves strong predictive accuracy for $W(G)$, $M_1(G)$, $M_2(G)$, and $R(G)$, with R^2 values ranging from 0.9716 to 0.9857 and original-scale MAPE values between 2.33% and 5.68%. However, its performance declines for the global combinatorial indices, reaching $R^2 = 0.4473$ and MAPE = 66.99% for $MS(G)$, and $R^2 = 0.6726$ and MAPE = 87.75% for $Z(G)$. This pattern is consistent with the theoretical distinction between degree- or distance-related targets and global counting functions. A three-layer GIN can capture useful local structural correlations, but its 1-WL-bounded message-passing representation cannot generally recover all independent-set and matching configurations. The stress-test results therefore indicate a potential expressivity bottleneck for targets that require global combinatorial counting. This interpretation does not constitute causal proof because the experiment does not compare GIN with higher-order architectures under matched conditions, and the observed performance decline may also reflect data coverage, optimization, and network depth. However, the stronger performance of MLP and Random Forest when supplied with informative global descriptors suggests that graph structure alone is insufficient when the learning architecture cannot represent all target-relevant distinctions.

The stress experiment therefore provides a conditional justification for graph-native surrogate prediction. GIN enables fast prediction directly from node features and sparse connectivity. However, descriptor-based nonlinear models remain preferable when the required graph summaries are available, and maximum predictive accuracy is the primary objective. Surrogate prediction is justified only when exact computation is sufficiently costly, the expected approximation error is acceptable, and the trained model is reused across enough structurally related graph queries to amortize the costs of exact labeling and model training.

Amortized Break-Even Analysis

A fair comparison between exact evaluation and surrogate prediction must include the one-time cost of generating exact training labels, the model-training cost, and the complete online prediction path. Let T_{label} denote the exact labeling time of the training set, T_{train} the model-training time, t_{exact} the average exact evaluation time per new graph, and t_{sur} the end-to-end surrogate time per graph, including graph preprocessing, descriptor or graph-tensor construction, and model inference. For Q future graph queries, the total exact-computation cost is

$$T_{\text{exact}}(Q) = Q t_{\text{exact}}, \quad (23)$$

whereas the total surrogate cost is

$$T_{\text{surrogate}}(Q) = T_{\text{label}} + T_{\text{train}} + Q t_{\text{sur}}. \quad (24)$$

The surrogate becomes computationally favorable when $T_{\text{surrogate}}(Q) < T_{\text{exact}}(Q)$. The corresponding break-even query count is

$$Q^* = \frac{T_{\text{label}} + T_{\text{train}}}{t_{\text{exact}} - t_{\text{sur}}}, \quad t_{\text{exact}} > t_{\text{sur}}. \quad (25)$$

For the original bounded unicyclic–bicyclic benchmark, $t_{\text{exact}} = 10.27$ ms per graph, and the reported GIN forward-pass time is 0.47 ms per graph. Although this corresponds to a relative speedup of approximately $21.9\times$, the absolute saving remains small because both exact and learned evaluations are completed within milliseconds. Consequently, the bounded benchmark does not, by itself, justify surrogate deployment on computational grounds.

For the larger sparse-multicyclic stress experiment, exact labeling of the 112 training graphs was required

$$T_{\text{label}} = 721.843 \text{ s},$$

and the mean exact evaluation time on the held-out test graphs was

$$t_{\text{exact}} = 6.529 \text{ s/graph}.$$

The end-to-end online times, including preprocessing and prediction, were 5.090 ms for MLP, 5.518 ms for GIN, and 143.425 ms for Random Forest. The corresponding model-training times were 1.627 s, 6.731 s, and 0.908 s, respectively. Table 25 summarizes the resulting amortized break-even values.

Table 25. End-to-end timing and amortized break-even analysis for the larger sparse-multicyclic stress experiment.

Model	Training-Label Time (s)	Model-Training Time (s)	End-to-End Time (ms/Graph)	Mean Speedup Over Exact	Break-Even Queries
MLP	721.843	1.627	5.090	$1282.6\times$	111
GIN	721.843	6.731	5.518	$1183.2\times$	112
Random Forest	721.843	0.908	143.425	$45.5\times$	114

For GIN, substitution into the break-even expression gives

$$Q_{\text{GIN}}^* = \frac{721.843 + 6.731}{6.529 - 0.005518} = 111.685, \quad (26)$$

which yields a conservative integer break-even threshold of 112 subsequent queries. The corresponding thresholds are 111 queries for MLP and 114 queries for Random Forest.

Table 25 reports the exact timing and amortized break-even values, while Figure 13 visually summarizes the corresponding offline and online computational trade-offs. Exact label generation dominates the offline cost for all models, which explains why the break-even thresholds remain similar despite substantial differences in end-to-end prediction time.

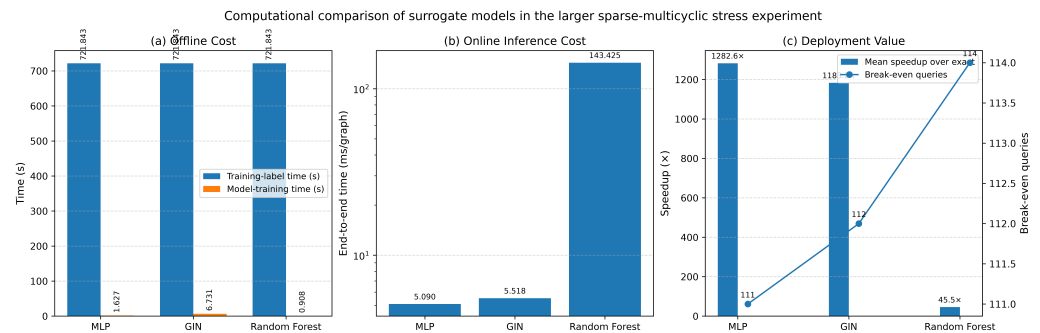


Figure 13. Computational comparison of the surrogate models: (a) offline cost, (b) end-to-end prediction time, and (c) speedup with amortized break-even queries.

These results show that all three learned surrogates become computationally favorable after approximately 111–114 future graph evaluations under the evaluated stress-test workload. MLP provides the strongest combination of aggregate predictive accuracy and end-to-end speed, whereas GIN offers a similarly low online cost while operating directly on a sparse graph. The Random Forest remains accurate but incurs a higher online cost because its descriptor-based prediction path is comparatively slower.

The break-even values are specific to the evaluated graph distribution, memory-isolated exact-labeling implementation, preprocessing procedures, hardware configuration, and model-training protocol. The exact evaluation time includes the isolated subprocess execution used for $MS(G)$ and $Z(G)$, and the reported surrogate times include complete end-to-end preprocessing and inference. The thresholds should therefore be interpreted as workload-specific operational estimates rather than universal complexity bounds.

5.11. Discussion of Observed Trends

The experimental results show four main trends. First, Random Forest and MLP achieve the strongest aggregate and per-index accuracy on the bounded unicyclic–bicyclic benchmark. This advantage comes from the use of engineered graph-level descriptors that encode graph order, degree distribution, branching behavior, and cycle complexity. Second, GIN consistently outperforms GCN and GAT, making it the strongest graph-native model, though it remains less accurate than the descriptor-based nonlinear predictors.

Third, target difficulty depends on the structural information required by each index. The Zagreb and Randić indices depend mainly on degree patterns, while the Wiener index is closely related to distance structure. These properties are either explicitly represented or strongly correlated with the available features. In contrast, $MS(G)$ and $Z(G)$ require global independent-set and matching counts, which explains their larger performance degradation in the sparse multicyclic stress experiment.

Fourth, the target-wise performance gap is consistent with the expressivity limits of first-order message-passing architectures. GIN can reach the discriminative power of the 1-WL test, but it cannot generally distinguish all graphs that differ in global combinatorial invariants. The results therefore indicate an accuracy–representation–expressivity trade-off. Descriptor-based models are preferable when informative global summaries are available,

whereas GIN offers a reusable graph-native formulation whose performance remains limited by the structural information captured through first-order message passing.

5.12. Limitations of the Predictive Framework

The benchmark uses synthetic unicyclic and bicyclic graphs, which enable controlled generation, exact labeling, and reproducible evaluation. However, these graph families do not fully represent the structural diversity of molecular, communication, biological, or infrastructure networks. Therefore, the reported results remain conditional on the distributions generated by the graph. In addition, the supervised framework requires exact labels during training, which limits its computational advantage for isolated small graphs whose indices can already be computed within milliseconds. Its practical value mainly depends on repeated inference over sufficiently large query workloads with acceptable approximation error.

The comparison between descriptor-based and graph-native models also reflects different input representations. Random Forest and MLP use informative graph-level descriptors, whereas GIN learns from sparse connectivity and node-level features. Thus, the results compare complete predictive pipelines rather than models trained from identical raw information. Moreover, GIN, GCN, and GAT remain first-order message-passing architectures and are subject to the general 1-WL expressivity limit. Structural node features enrich the input but do not remove this limitation. Since higher-order, subgraph-based, or k -WL-style models were not evaluated, the weaker performance on $MS(G)$ and $Z(G)$ is consistent with an expressivity bottleneck but does not prove it as the only cause.

The external circulant experiment covers only one highly regular graph family. It demonstrates pipeline portability in a structured setting but does not establish transferability to arbitrary graph classes. Similarly, the larger multicyclic experiment includes only instances that could be exactly labeled within the prescribed runtime budget. As a result, the predictive results may favor computationally easier graphs and cannot be directly extended to timeout cases. Generalization to larger cycle ranks, weighted or directed graphs, new index families, and unseen structural distributions remains an open direction.

6. Conclusions and Future Work

This study presented a reproducible framework integrating controlled cyclic graph generation, structure-preserving transformations, exact computation of six topological indices, sparse graph representation, and multi-output regression. Descriptor-based non-linear models achieved the lowest aggregate errors on the bounded unicyclic and bicyclic benchmarks because the supplied graph-level summaries encode strong prior structural information. GIN did not outperform MLP or Random Forest in overall accuracy, but it remained the strongest graph-native architecture and learned directly from sparse connectivity and node-level features without requiring a fixed handcrafted descriptor vector. Its weaker performance on $MS(G)$ and $Z(G)$ is consistent with the limited ability of 1-WL-bounded message-passing architectures to represent global combinatorial counting functions, although matched higher-order experiments are required to distinguish this theoretical limitation from optimization and data effects. Exact computation remained preferable for isolated graphs with $n \leq 30$ because both exact and learned evaluations operated at a millisecond scale. The expanded circulant experiment, covering 200 graphs across generator dimensions $k \in 2, 3, 4$ with repeated splits and an order-held-out evaluation, provided statistically stronger evidence that the downstream pipeline can be transferred to an external structured graph family by modifying only the graph-construction module. In contrast, the sparse-multicyclic stress experiment showed that surrogate inference becomes relevant only when exact computation is sufficiently expensive, predictive accuracy is ac-

ceptable, and labeling and training costs can be amortized over a large query workload. The framework should therefore be viewed as a workload-dependent approximation strategy rather than a replacement for exact graph algorithms.

Future work should extend the evaluation to larger and more diverse, exactly labeled graph collections, incorporate specialized counting methods, and study uncertainty and out-of-distribution behavior. A controlled expressivity analysis should compare the current GIN with deeper parameter-matched GINs, positional-encoding variants, substructure-augmented models, subgraph GNNs, and higher-order or k -WL-style architectures under identical splits and training budgets, especially for $MS(G)$ and $Z(G)$. These extensions should also report time and memory costs, since more expressive graph representations may introduce substantial computational overhead.

Author Contributions: Conceptualization, N.K. and Y.I.; methodology, N.K., M.Z. and M.I.; software, M.A.I. and M.I.; validation, S.A., Y.I. and M.Z.; formal analysis, N.K. and S.A.; investigation, N.K., M.I. and M.A.I.; resources, S.A. and M.Z.; data curation, M.A.I. and S.A.; writing—original draft preparation, N.K. and M.I.; writing—review and editing, Y.I., M.A.I. and S.A.; visualization, M.Z. and Y.I.; supervision, Y.I. and S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The code is available at: [38].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zheng, W.; Xu, G.; Lu, S.; Lyu, J.; Bao, F.; Yin, L. GNN: Core Branches, Integration Strategies and Applications. *Comput. Model. Eng. Sci.* **2026**, *146*. [CrossRef]
2. Lyu, J.; Xu, G.; Bao, F.; Zhou, Y.; Liu, Y.; Lu, S.; Zheng, W. Multi-label classification model using graph convolutional neural network for social network nodes. *Comput. Model. Eng. Sci.* **2026**, *146*. [CrossRef]
3. Zheng, S.; Liu, Z.; Zhu, Z.; Zhang, X.; Li, J.; Zhao, Y. BiKT: Unleashing the potential of GNNs via Bi-directional Knowledge Transfer. *IEEE Trans. Pattern Anal. Mach. Intell.* **2025**, *48*, 3304–3318.
4. Munir, K.; Khan, N.; Iqbal, M.; Shebka, N.; Alahmari, S.; Shakir, D. Scaling Unicyclic Graph Analysis: Automated Generation, Augmentation, and Machine Learning-Driven Prediction. In *Proceedings of the 2024 International Conference on Engineering and Emerging Technologies (ICEET)*; IEEE: New York, NY, USA, 2024; pp. 1–6.
5. Hong, Y.; Liu, H. On the Wiener Index of Unicyclic Graphs ABSTRACT | FULL TEXT. *Hacet. J. Math. Stat.* **2011**, *40*, 63–68.
6. Chen, Y.H.; Zhang, X.D. The Wiener index of unicyclic graphs with girth and the matching number. *arXiv* **2011**, arXiv:1102.5409.
7. Deng, H. A unified approach to the extremal Zagreb indices for trees, unicyclic graphs and bicyclic graphs. *MATCH Commun. Math. Comput. Chem.* **2007**, *57*, 597–616.
8. Li, S.; Zhao, Q. Sharp upper bounds on Zagreb indices of bicyclic graphs with a given matching number. *Math. Comput. Model.* **2011**, *54*, 2869–2879. [CrossRef]
9. Wang, H.; Xu, K. The minimum Wiener index of Halin graphs with characteristic trees of diameter 4. *Electron. J. Math.* **2025**, *9*, 11–22. [CrossRef]
10. Khan, N.; Shamus, M.; Hussain, F.G.; Iqbal, M. Minimum Degree Distance of Five Cyclic Graphs. *Pure Appl. Math. J.* **2021**, *10*, 84–88. [CrossRef]
11. Yang, B.; Manjalapur, V.V.; Sajjan, S.P.; Mathai, M.M.; Liu, J.B. On extended adjacency index with respect to acyclic, unicyclic and bicyclic graphs. *Mathematics* **2019**, *7*, 652. [CrossRef]
12. Wiener, H. Structural determination of paraffin boiling points. *J. Am. Chem. Soc.* **1947**, *69*, 17–20. [CrossRef] [PubMed]
13. Randić, M. Characterization of molecular branching. *J. Am. Chem. Soc.* **1975**, *97*, 6609–6615. [CrossRef]
14. Gozalbes, R.; Doucet, J.; Derouin, F. Application of topological descriptors in QSAR and drug design: History and new trends. *Curr. Drug-Targets-Infect. Disord.* **2002**, *2*, 93–102. [CrossRef] [PubMed]
15. Todeschini, R.; Consonni, V. *Molecular Descriptors for Chemoinformatics: Volume I: Alphabetical Listing/Volume II: Appendices, References*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
16. Amigó, J.M.; Gálvez, J.; Villar, V.M. A review on molecular topology: Applying graph theory to drug discovery and design. *Naturwissenschaften* **2009**, *96*, 749–761. [CrossRef] [PubMed]

17. Arul, S.M.; Senthil, G.; Jayasudha, S.; Alkhayyat, A.; Azam, K.; Elangovan, R. Graph theory and algorithms for network analysis. In *Proceedings of the E3S Web of Conferences*; EDP Sciences: Les Ulis, France, 2023; Volume 399, p. 08002.
18. Mahmood, M.K.; Ahmad, D. Order structured graphs of cyclic groups and their classification. *VFAST Trans. Math.* **2024**, *12*, 220–233. [[CrossRef](#)]
19. Romanov, A.Y. Development of Routing Algorithms in Networks-on-Chip Based on Ring Circulant Topologies. *Heliyon* **2019**, *5*, e01516. [[CrossRef](#)] [[PubMed](#)]
20. Romanov, A.Y.; Lezhnev, E.V.; Glukhikh, A.Y.; Amerikanov, A.A. Development of Routing Algorithms in Networks-on-Chip Based on Two-Dimensional Optimal Circulant Topologies. *Heliyon* **2020**, *6*, e03183. [[CrossRef](#)] [[PubMed](#)]
21. Romanov, A.Y.; Myachin, N.M.; Lezhnev, E.V.; Ivannikov, A.D.; El-Mesady, A. Ring-Split: Deadlock-Free Routing Algorithm for Circulant Networks-on-Chip. *Micromachines* **2023**, *14*, 141. [[CrossRef](#)] [[PubMed](#)]
22. Monakhova, E.A.; Romanov, A.Y.; Lezhnev, E.V. Shortest Path Search Algorithm in Optimal Two-Dimensional Circulant Networks: Implementation for Networks-on-Chip. *IEEE Access* **2020**, *8*, 215010–215019. [[CrossRef](#)]
23. Shchegoleva, M.A.; Romanov, A.Y.; Lezhnev, E.V.; Amerikanov, A.A. Routing in Networks on Chip with Multiplicative Circulant Topology. *J. Phys. Conf. Ser.* **2019**, *1163*, 012027. [[CrossRef](#)]
24. Myachin, D.A.; Pugach, V.P.; Avdeyuk, S.S.; Zunin, V.V.; Romanov, A.Y. The Open System for Storing and Processing of a Dataset of Combinational Circuits. *Proc. Inst. Syst. Program. RAS* **2023**, *35*, 81–90. [[CrossRef](#)] [[PubMed](#)]
25. Zunin, V.; Romanov, A.Y.; Solovyev, R. Developing methods for combinational circuit generation. In *Proceedings of the 2022 International Russian Automation Conference (RusAutoCon)*; IEEE: New York, NY, USA, 2022; pp. 842–846.
26. Zhang, H.; Lu, G.; Zhan, M.; Zhang, B. Semi-supervised classification of graph convolutional networks with Laplacian rank constraints. *Neural Process. Lett.* **2022**, *54*, 2645–2656. [[CrossRef](#)]
27. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. In *Proceedings of the International Conference on Learning Representations*; ICLR: Ithaca, NY, USA, 2018; Volume 6.
28. Li, P.; Leskovec, J. The expressive power of graph neural networks. In *Graph Neural Networks: Foundations, Frontiers, and Applications*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 63–98.
29. Morris, C.; Lipman, Y.; Maron, H.; Rieck, B.; Kriege, N.M.; Grohe, M.; Fey, M.; Borgwardt, K. Weisfeiler and leman go machine learning: The story so far. *J. Mach. Learn. Res.* **2023**, *24*, 1–59.
30. Liang, M.; Zhang, H. TAGNN: Topology-aware graph neural network framework for link prediction. *Sci. Rep.* **2026**, *16*, 17292. [[CrossRef](#)] [[PubMed](#)]
31. Jegelka, S. Theory of graph neural networks: Representation and learning. In *Proceedings of the International Congress of Mathematicians*; European Mathematical Society-EMS-Publishing House GmbH: Berlin, Germany, 2023; pp. 5450–5476.
32. Besta, M.; Scheidl, F.; Gianinazzi, L.; Kwasniewski, G.; Klaiman, S.; Müller, J.; Hoefler, T. Demystifying higher-order graph neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2025**, *48*, 2544–2565. [[CrossRef](#)] [[PubMed](#)]
33. Egressy, B.; Von Niederhäusern, L.; Blanuša, J.; Altman, E.; Wattenhofer, R.; Atasu, K. Provably powerful graph neural networks for directed multigraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*; Association for the Advancement of Artificial Intelligence: Washington, DC, USA, 2024; Volume 38, pp. 11838–11846.
34. Kanatsoulis, C.I.; Ribeiro, A. Graph neural networks are more powerful than we think. In *Proceedings of the ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; IEEE: New York, NY, USA, 2024; pp. 7550–7554.
35. Zhao, T.; Jin, W.; Liu, Y.; Wang, Y.; Liu, G.; Günemann, S.; Shah, N.; Jiang, M. Graph data augmentation for graph machine learning: A survey. *arXiv* **2022**, arXiv:2202.08871.
36. Romanov, A. Cyclic Graph Index Prediction Benchmark. GitHub Repository. 2022. Available online: <https://github.com/RomeoMe5/circulantGraphs> (accessed on 24 June 2026).
37. Romanov, A. The Dataset for Optimal Circulant Topologies. *Big Data Cogn. Comput.* **2023**, *7*, 80. [[CrossRef](#)]
38. Iqbal, M. Cyclic Graph Index Prediction Benchmark. GitHub Repository. 2026. Available online: <https://github.com/iqbalmansoor1986-hue/Cyclic-Graph-Index-Prediction-Benchmark> (accessed on 21 June 2026).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.