1.5em 0pt

2FWL-SIRGN: A Scalable Structural 2-dimensional Folklore Weisfeiler Lehman Graph Representation Learning Approach Via Structural Graph Partitioning

Justin Carpenter
Computer Science Dept.
Boise State University
Boise, Idaho, United States
JustinCarpenter836@u.boisestate.edu

Edoardo Serra
Computer Science Dept.
Boise State University
Boise, Idaho, United States
edoardoserra@boisestate.edu

***Abstract*—Graph representation learning has numerous applications, ranging from social networks to bioinformatics, with a major focus on Graph Neural Networks (GNNs). However, many GNN models face challenges in capturing intricate graph structures, such as cycles, and are prone to overfitting and high computational costs, limiting their scalability on medium to big graphs.**

In this paper, we propose 2FWL-SIRGN, a novel approach that integrates higher-order Weisfeiler-Lehman (WL) test algorithm while mitigating its computational challenges. Our method combines the Structural Iterative Representation Learning for Graph Nodes (SIRGN) framework with the 2-dimensional Folklore Weisfeiler-Lehman (2FWL) isomorphism test. The unsupervised training of the SIRGN component improves the model's resistance to overfitting, while the 2FWL component enhances its expressive power, enabling it to capture complex patterns, such as cycle structures. However, the inclusion of 2FWL increases computational overhead. To address this, we introduce a Structural Graph Partitioning algorithm, which allows 2FWL-SIRGN to scale efficiently to big graphs.

Extensive experiments demonstrate that 2FWL-SIRGN outperforms state-of-the-art methods by addressing key challenges in graph representation learning. Our model captures richer structural information while maintaining computational efficiency, surpassing other higher-order WL approaches. Additionally, our partitioning strategy enables 2FWL-SIRGN to effectively handle large-scale graphs, and its inherent resistance to overfitting addresses a common limitation of GNNs. These advancements position 2FWL-SIRGN as a robust solution for real-world applications where both scalability and accuracy are critical.

***Index Terms*—Graph Representation Learning, Weisfeiler-Lehman Isomorphism Test, SIR-GN, Structural Graph Partitioning**

I. Introduction

Graphs are powerful data structures that represent entities as nodes and the relationships between them as edges. The graph-structured data is one of the most ubiquitous forms of structured data used in machine learning approaches such as graph neural networks (GNN) and graph representation learning alike [1]. In order to utilize graph-structured data, many studies have been done to transform these complex structures into a format that can be easily processed by machine learning algorithms while preserving the graph's informational content.

Recent years have seen a surge in research on graph representation learning techniques in various machine learning tasks such as node classification, link prediction, and graph classification. Instead of extracting hand-engineered features, graph representation learning aims to learn representations that encode structural information about the graph [2].

There are two well explored concepts within graph representation learning, structural and proximity, each with the same end goal to transform a graphs informational content into a usable format. Although similar end goal, the two concepts differ from one another on capturing different types of information within the graph. Structural representation learning focuses on capturing the positions of nodes within the entire graph, irrespective of their local neighborhoods. The goal to identify nodes that have similar structures and preserver the overall graph topology. Proximity representation learning emphasizes individual node neighbors or proximity to a centroid in order to preserve the local neighborhood structure of nodes [3]–[5].

Traditional proximity-based representation learning methods, such as node2vec [4] and DeepWalk [3], optimize embeddings to encode the statistics of random walks to define node similarity and neighborhood reconstruction. The problem with both approaches was the ability to capture the graphs structure and complexity.

Structural graph representation learning approaches such as, Structural Iterative Representation Learning Approach for Graph Nodes (SIR-GN) [6] and Graph Isomorphism Network (GIN) [7] attempt to more accurately capture the graph structure than previous proximity based approaches. Such approaches simulates the WL test [8] which is an approximated algorithm for verifying that two graphs are isionmorphic. These approaches can distinguish almost all pairs of isomorphic graphs. Their major limitation is their inability to distinguish non-isomorphic graphs with cycles of varying lengths.

To overcome such limitation graph neural networks simulating higher order WL test have been prosed [9]–[11]. The problem that arises when implementing higher-order WL is the increase in computational cost [8] and an increased risk of overfitting.

In this paper, we present 2FWL-SIRGN, a novel approach that integrates the higher-order Weisfeiler-Lehman (WL) test algorithm while addressing its computational challenges. Our method combines the Structural Iterative Representation Learning for Graph Nodes (SIRGN) framework with the 2-dimensional Folklore Weisfeiler-Lehman (2FWL) isomorphism test which is higth order WL isomorphic test. The unsupervised training of the SIRGN component enhances the model's ability to mitigating overfitting, while the 2FWL component increases its expressive power, enabling it to capture complex patterns like cycle structures. To counter the increased computational load from 2FWL, we introduce a Structural Graph Partitioning algorithm, allowing 2FWL-SIRGN to efficiently scale to large graphs. In the following we present the **List of our contributions:**

1) **2FWL-SIRGN**: Design of a higher-order WL test SIRGN.
2) **Structural Graph Partition**: Design of a structural graph partitioning algorithm to overcome te computational cost of 2FWL-SIRGN.
3) **Experimentation and Validation**: The implementation of tests designed to evaluate structural capabilities of our proposed algorithm and existing methods.

II. Related Works

In the literature, many graph representation approaches are closely linked to the Weisfeiler-Lehman (WL) isomorphism test, a heuristic algorithm designed to determine whether two graphs are isomorphic. In this section, we first provide an overview of the WL isomorphism test and its higher-order variants. We then explore the graph representation learning procedures associated with these methods

A. (Folklore) Weisfeiler-Lehman Isomorphism Test

The Weisfeiler-Lehman (WL) Isomorphism Test is a heuristic algorithm widely used to determine whether two graphs are isomorphic. The graph isomorphism problem remains a significant challenge in computational theory, as no polynomial-time algorithm has yet been discovered. As a heuristic, the WL test can effectively distinguish many pairs of graphs as non-isomorphic. However, its output is binary: either "non-isomorphic" or "possibly isomorphic," meaning it cannot confirm that two graphs are truly isomorphic.

The 1-WL Test assigns the same label to each node and iteratively refines this label based on the labels of neighboring nodes. The algorithm reaches convergence when the distribution of the labels across the nodes ceases to change. At this point, if the label distributions of two graphs are dissimilar, the graphs are deemed non-isomorphic; if similar, they may be isomorphic. This method falls within the category of message-passing algorithms and is known for its linear execution time, at each iteration, relative to the graph's edge count, and linear space complexity relative to the node count [8], [12].

The K-WL method extends beyond assigning labels to individual nodes, instead allocating identifiers to each K-tuple of nodes within V^K of graph G. With this extension, the notion of a node's neighborhood expands to encompass the neighborhoods of a K-tuple, increasing both the isomorphism test's execution time and space complexity.

Algorithm 1 k-dimensional Folklore WL (kFWL)

Require: Graphs $G_1 = (V_1, E_1, X^1)$ and $G_2 = (V_2, E_2, X^2)$

$(v_0, v_1, \ldots, v_{k1}), [i] \leftarrow w = (v_1, \ldots, v_{i1}, w, v_{i+1}, \ldots v_{k1})$

$c_v^{(0)} = HASH(< X_a^1 | a \in v >, < (i,j) | i,j \in \{0,\ldots,k1\}, (v[i], v[j]) \in E_1 >) \quad \forall v \in (V_1)^k$

$d_v^{(0)} = HASH(< X_a^2 | a \in v >, < (i,j) | i,j \in \{0,\ldots,k1\}, (v[i], v[j]) \in E_2 >) \quad \forall v \in (V_2)^k$

for $l = 1, 2, \ldots$ **do**

  if $\{\{c_v^{(i1)} | v \in V_1^k\}\} \neq \{\{d_v^{(i1)} | v \in V_2^k\}\}$ **then**

    return "Non-Isomorphic"

  end if

  $c_{v,w}^{(l)} = (c_{v[0]\leftarrow w}^{(l1)}, c_{v[1]\leftarrow w}^{(l1)}, \ldots, c_{v[k1]\leftarrow w}^{(l1)}) \quad \forall v \in (V_1)^k, w \in V_1$

  $c_v^{(l)} = HASH(c_v^{(l1)}, \{\{c_{v,w}^{(l1)} | w \in V_1\}\}) \quad \forall v \in (V_1)^k$

  $d_{v,w}^{(l)} = (d_{v[0]\leftarrow w}^{(l1)}, d_{v[1]\leftarrow w}^{(l1)}, \ldots, d_{v[k1]\leftarrow w}^{(l1)}) \quad \forall v \in (V_2)^k, w \in V_2$

  $d_v^{(l)} = HASH(d_v^{(l1)}, \{\{d_{v,w}^{(l1)} | w \in V_2\}\}) \quad \forall v \in (V_2)^k$

end for

return "Possibly Isomorphic"

The K-WL establishes a hierarchy where, for any $K > 2$, the $(K+1)$-WL is strictly more expressive than the K-WL, with an equivalence noted between 2-WL and 1-WL. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the cost of K-WL at each iteration is $O(|V_1|^{K+1} + |V_2|^{K+1})$. Since the first method with higher expressive power than 1-WL is 3-WL, the cost of each iteration is $O(|V_1|^4 + |V_2|^4)$, which is impractical for medium and large graphs.

To improve computational complexity, the Folklore Weisfeiler-Lehman (FWL) test was introduced.

The Folklore Weisfeiler-Lehman (FWL) test, shown in Algorithm 1, is a variant of the Weisfeiler-Lehman (WL) test that offers improved efficiency. The FWL test works by considering k-tuples of nodes, similar to the k-WL test, but with a slightly modified update rule.

The 2-dimensional FWL (2FWL) test is as powerful as the 3-dimensional WL (3WL) test. The cost of the 2FWL at each iteration is $O(n^3)$, while the cost of the 3-dimensional WL test is $O(n^4)$ [13]. Despite the improvement in complexity, the 2FWL's complexity remains infeasible for medium and large graphs

B. WL-related Graph Representation Learning Techniques

The expressive capabilities of GNNs in relation to the WL test have been extensively studied. Standard Graph Neural Networks (GNNs), such as Graph Convolutional Networks (GCNs) [14] and Graph Attention Networks (GATs) [15], share some algorithmic similarities with the 1-WL test. However, they generally fail to fully match its expressive power.

[16] introduced a method for learning node representations using GCNs, employing the WL test to assess the expressive power of GCNs compared to traditional graph-based learning techniques. The study demonstrated that GCNs could capture more expressive representations than conventional approaches such as sequence-to-sequence learning [17].

Similarly, [18] proposed an approach for learning node representations in structured data using graph CNNs. By leveraging the WL test, the authors showed that their method achieved more expressive representations than traditional techniques based on recursive neural networks.

Other studies have also explored node representation learning in social networks. For example, [3] proposed the DeepWalk algorithm, which utilizes random walks to learn embeddings, using the WL test to confirm its ability to effectively capture structural properties of social networks. Additionally, [19] developed a method that incorporates biased random walks for enhanced node representation in networks, demonstrating through the WL test that their approach successfully captured complex structural patterns.

Spectral approaches have also been explored in the context of the WL test. [20] introduced a method for node representation learning using spectral networks and locally connected architectures. The WL test was employed to evaluate its expressive power, highlighting the advantages of this approach over traditional graph-based learning techniques.

Among GNNs, Graph Isomorphism Networks (GINs) stand out for their theoretical connection to the WL test. As shown in [7], GINs can achieve expressive power equivalent to the 1-WL test, with a direct relationship between the node representations generated by GINs and the node colors assigned by the WL algorithm. However, this expressive equivalence relies on the model achieving an optimal parameter set during training.

While the 1-WL test forms the foundation for many GNN architectures, more advanced graph representation learning techniques aim to increase expressive power by leveraging higher-order WL tests. For example, [11] introduced higher-order graph neural networks that utilize the k-WL test to capture more complex structural patterns. These methods significantly enhance representational power, enabling the detection of more nuanced graph structures. As another example, [21] describes a novel higher-order Weisfeiler-Lehman graph convolution network based on 2-FWL test.

Despite their promise, techniques based on higher-order WL tests face significant limitations, particularly in computational cost. These methods become impractical when applied to medium or large-scale graphs.

III. Methodology

The proposed solution in this paper is a novel graph representation learning model 2FWL-SIRGN, that incorporates the 2-dimensional Folklore Weisfeiler-Lehman (2FWL) test with Structural Iterative Representation Learning Approach for Graph Nodes (SIR-GN), coupled with a structural graph partitioning algorithm. This approach aims to accurately capture complex graph structures and provides scalability with larger graphs. In the following we provide the

1) **2FWL-SIRGN**: The proposed higher-order WL algorithm 4 design and cost analysis.
2) **Structural Graph Partitioning**: The proposed Structural Graph Partition algorithm 3 design and cost analysis.
3) **2FWL-SIRGN Utilizing Graph Partition**: Implementation of Structural Graph Partition

Algorithm 2 2FWL sirgn

function KMeansNodeDesc($V, PRNorm$)
  $CC = KMeans(PRNorm)$ ▷ Clustering step
  for all $u \in V$ **do** ▷ Vertex description loop
    $dV_u = CalcDist(PRNorm_u, CC)$
    $DV_u = (Max(dV_u) - dV_u)/(Max(dV_u) - Min(dV_u))$
    $DV_u = DV_u/Sum(DV_u)$
  end for
  return DV
end function
function 2FWL SIR-GN(G,d)
  $i = 0$
  Initialize a matrix $Emb \in \Re^{|V|^2 \times n^2}$
  for all $u \in V$ **do**
    for all $e \in nbr(u)$ **do**
      $Emb_{ue|V|}$ = $(0, 1, 0$ to $n-2)$
    end for
  end for
  while $i <= d$ **do**
    $PRNorm = MinMaxNorm(Emb^i)$
    $DV^i = $ KMeansNodeDesc$(V, PRNorm)$
    Initialize a matrix $Emb2 \in \Re^{|V|^2 \times 1}$ to 0
    for all $u_i \in V$ **do**
      for all $u_j \in V$ **do**
        for all $z \in nbr(u_i) \cup nbr(u_j)$ **do**
          $Emb_{u_i,u_j} = Emb_{u_i,u_j} + DV^i_{z,u_j}(DV^i_{u_i,z})^T$
        end for
      end for
    end for
    $i =$
  end while
end function

A. 2FWL-SIRGN

The first part of our proposed solution is the design and implementation of the 2FWL-SIRGN algorithm 4, an extension of the Structural Iterative Representation Learning Approach for Graph Nodes (SIR-GN) that emulates the 2-dimensional Folklore Weisfeiler-Lehman (2FWL) algorithm. The main objective is to create a more informative graph representation than the original SIR-GN. Our design involves the following steps:

Theorem 1. *2FWL-SIRGN cost* $O(N_p * n^2(|V_p|^2 * G_i + |V_p|^3))$

Proof. To prove Theorem 1, we start by looking at Algorithm 4, starting with the matrix initialization loop beginning on line 5 is $O(|V|^2 * n^2 + |E| * n)$. This is then followed by our computation in each iteration of graph partitions after line 13. We must normalize which on line 20 will cost $O(|V|^2 * n^2)$ then second matrix initalization on line 21 will cost $O(|V|^2 * n^2)$ finally out loops in range of line 22-27 will be of a cost $O(2 * |V| * |E| * n^2)$ for the final cost of $O(N_p * n^2(|V_p|^2 * G_i + |V_p|^3))$ □

Structural Graph Representational Learning (SGRL) is a machine learning technique that is used to learn representations of graph-structured data, such as social networks, protein interactions, and computer networks. The goal of SGRL is to learn a mapping from the nodes and edges in a graph to a low-dimensional space where they can be analyzed and manipulated. SGRL can be used in a variety of applications, such as node classification and link prediction. For example, in node classification, SGRL can be used to predict the class of a node in a graph based on its local neighborhood. In link prediction, SGRL can be used to predict the existence of an edge between two nodes in a graph.

B. Structural Graph Partitioning

To address the increased computational cost of the 2FWL-SIRGN method, we introduce a structural graph partitioning algorithm. This algorithm partitions large graphs into smaller, manageable subgraphs, while retaining the graph's structural information. This will allow SIR-GN to be computed on each subgraph, reducing the complexity of running on the entire graph. The steps include:

1) **Clustering**: Group nodes based on structural similarity using SIR-GN.
2) **Ranking**: Rank the groups using the PageRank algorithm to prioritize important structures.
3) **Partitioning**: Divide the graph into partitions based on the ranked groups, ensuring minimal loss of structural information.

While there are many graph partitioning approaches, including clustering [22], page rank [23], and cutting the graph at random [24]. These approaches cause a significant loss in graph structure information. **The main focus for structural graph partitioning:**

- **1.** How do we reduce the lost structural information of the graph?
- **2.** How do we partition the graph into a set number of partitions?
- **3.** Is this applicable for connected and disconnected graphs?

In order to reduce the loss of structural information of the graph, we utilized SIR-GN for it's fast runtime and strong structural information retained. Using SIR-GN we were able to select all the nodes within a graph with similar structures. This allowed us to group nodes and edges without losing critical structural information. Then by utilizing the page rank algorithm in order to rank our nodes in the groups and furthermore rank our groups. This allows for the lowest structural information loss we can achieve. Now we have our structure groups and them scored by importance, and we use a binary search to add edges to create our graphs for each partition we want. When adding edges, we consider the highest-ranked value edges first so we can retain as much of our original graphs structure. This is also applicable for fully connected graphs as we will in theory lose the lowest ranked edges for our groups, which should not impact any of the structural importance we collected from our SIR-GN run.

To retain structural information while partitioning a graph designed Algorithm 3 Structural Graph Partition. The algorithm commences by invoking the CreateClusters function, which takes into account sorted groups, graph indices ($gind$), the graph G, and a target size for the clusters. The function recursively divides the graph into left and right list of graphs until the component component count aligns with the target size. Concurrently, the algorithm employs the PageRank algorithm to calculate group scores, which are then sorted to facilitate the cluster creation process. The final output comprises subgraphs that are constructed based on these clusters. This algorithm is pivotal for scenarios requiring the partitioning of large-scale graphs into manageable subgraphs, thereby enhancing computational efficiency and facilitating more focused analyses.

C. 2FWL-SIRGN Utilizing Graph Partition

Through Structural Graph Partition Algorithm 3 we were able to effectively split the entirety of the graph into smaller subgraphs. This allows us to run the 2FWL-SIRGN on individual subgraphs, resulting in a smaller embedding matrix than the sizes of nodes squared. This allows us to retain the information of most edges, without a large amount of possible edges that do not currently exist. This results in a set of embeddings for each graph partition. Each embedding is the size of nodes squared (n^2), which makes the set of embedding matrices the size of each partition's nodes squared added together ($n_{g_1}^2 + n_{g_2}^2 + ... + n_{g_i}^2$). This is a drastically reduced computational size then the original size of n^2.

The decrease in smaller embedding matrix id due to each iteration we compute the list of embeddings of each partition. Each partition only needs the information of connected components to effectively apply the algorithm in comparison of the graphs entirety. This allows our matrix per iteration to remain the size of the partition's nodes squared. 2FWL-SIRGN originally stored and processed every edge that could be possible, resulting in a large number of zeros due to nodes never having an edge with another. In our experiments we can validate this is acceptable for our research and in many cases reduce the chance of over-fitting which improves the accuracy by over 10%. II

Since each subgraph is smaller, we expect the total node complexity to be significantly less than processing the entire graph at once. Specifically, if the function's complexity is loglinear, we can expect that $|V|^2 >> \sum_{i=1}^{p} |V_i|^2$ where $||V||$ is the number of nodes in the original graph and $|V_i|$

Algorithm 3 Structural Graph Partition

function CREATECLUSTERS(sorted_groups, gind, G, target_size)
  $lower, upper = 0$
  while $lower > upper$ **do**
    $middle = round((lower + upper)/2)$
    $left\_group = sourted\_group[: middle]$
    $cc = ConnectedComponents(left\_group)$
    $maxNode = max_{a \in cc}|V_a|$
    if $maxNode \leq target\_size$ **then**
      $lower = middle$
    else
      $upper = middle$
    end if
  end while
  $left\_group = sourted\_group[: lower]$
  $cc = ConnectedComponents(left\_group)$
  return cc
end function
function CALCULATEGROUPSCORES(emb2, page_rank)
  $hashgroupscore \leftarrow \{\}, hashgroupind \leftarrow \{\}$
  for $group$ in $emb2$ **do**
    $score \leftarrow 0, nodes \leftarrow []$
    for $node$ in $group$ **do**
      $score \mathrel{+}= page\_rank[node]$
      $nodes$.append($node$)
    end for
    $hashgroupscore[group] \leftarrow score$
    $hashgroupind[group] \leftarrow nodes$
  end for
  return $hashgroupscore, hashgroupind$
end function
function PARTITIONGRAPH(G, n, num_partitions)
  $siremb \leftarrow$ **SirGN**(G, n)
  $emb2 \leftarrow$ **GroupEdges**($siremb$)
  $target\_size \leftarrow$ **TargetGroupSize**(G, $num\_partitions$)
  $page\_rank \leftarrow$ **PageRank**(G)
  $groupscore, groupind \leftarrow$ **CalculateGroupScores**($emb2$, $page\_rank$)
  $sorted\_groups \leftarrow$ **SortGroups**($groupscore$)
  $clusters \leftarrow$ **CreateClusters**($sorted\_groups$, $groupind$, G, $target\_size$)
  $subgraphs \leftarrow$ **ConstructSubgraphs**($clusters$, $groupind$)
  return $subgraphs$
end function

is the total number of nodes in the given partition of i. The total complexity of our proposed Structural Graph Partition algorithm is $O(E \log E)$ where $|E|$ is the number of edges.

The efficiency gain arises from the fact that processing smaller subgraphs separately can avoid the combinatorial explosion of possibilities that one needs to consider when processing the entire graph as a whole.

IV. EXPERIMENT AND RESULTS

A. Datasets

The datasets chosen for our experiments are well known in the fields of graph learning with baseline truths already provided and have a wide range in sizes. This is beneficial for testing the optimization of our methods and compare against the related works. The datasets Mutag, Enzyme, PTC, FM, NCI1, NCI109, Proteins, IMDb binary, and IMDb multi have gained prominence as essential benchmarks for evaluating the efficacy and performance of various graph classification methods. Graph classification, a central area of investigation within the area of graph representation learning, involves categorizing graphs into predefined classes or categories. The datasets encompass a diverse range of application domains, including chemical compounds, protein structures, and movie databases, thereby encompassing a broad spectrum of real-world scenarios. The datasets' influence is accentuated by their widespread adoption within notable institutions such as Texas University, Cornell University, and Wisconsin University. Researchers at these academic hubs have harnessed the datasets as foundational elements in their explorations of graph classification techniques. The datasets have contributed significantly to studies aiming to uncover the intricate relationships between graph structures and classification outcomes.

B. Experimental Setup

We evaluated our proposed 2FWL-SIRGN model using several well-known datasets in the field of graph learning, including MUTAG [25], PTC [26], NCI1 [27], NCI109 [28],

Algorithm 4 2FWL-SIRGN Algorithm

function 2FWL-SIRGN($G, n, nl, iter$)
  $SubG \leftarrow PartitionGraph(G, n, p)$ ▷ Create p number of graph partitions
  $emb\_list \leftarrow List[p]$
  $nodeLabels \leftarrow List[p]$
  for G_i in range($SubG$) **do** ▷ Initialize list of partitions variables
    $nv \leftarrow Size(SubG[G_i][node_1])$ ▷ Initialize nv
    $emb \leftarrow (nv \times nv) \times (n \times n)$ ▷ Initialize emb
  end for
  for i in range($iter$) **do**
    for G_i in range($SubG$) **do**
      $nv \leftarrow Size(SubG[G_i][node_1])$
      $count \leftarrow getnumber(emb\_list[G_i])$
      $emb1 \leftarrow Normalize(emb\_list[G_i])$
      $val \leftarrow PCA(n/2, emb1)$ ▷ Perform PCA on $emb1$ with $n/2$ components
      Let $val \leftarrow [val, -val]$ ▷ Append the negation of val to itself horizontally
      $subx \leftarrow Normalize(val)$ ▷ Normalize rows of val
      $emb2 \leftarrow (nv \times nv) \times (n \times n)$ ▷ Initialized to zeros
      for x in range(nv) **do**
        for y in range(nv) **do**
          $intersetz \leftarrow getIntersection(G[x].keys(), G[y].keys())$
          for z in $intersetz$ **do**
            if z is a key in G **then**
              $emb2[x \times nv + y] = subx[nv \times z + y] \times subx[nv \times x + z]$
            end if
          end for
        end for
      end for
      Compute $newCount \leftarrow getnumber(emb2)$
      if $count \geq newCount$ **then**
        break
      else
        Update $count$ to $newCount$
        Stack $emb2$ and $labtotal$ horizontally to update emb
      end if
    end for
  end for
  return the submatrix of emb corresponding to the diagonal elements
end function

TABLE I: Details about the datasets used in experiments

Method	Nodes	Node Classes	Edges	Graphs	Graph Classes
Mutag	3371	7	7442	188	2
Enzymes	19580	3	74564	600	6
PTC_FM	4925	18	10110	349	2
NCI1	122747	37	265506	4110	2
NCI109	122494	38	265208	4127	2
Proteins	43471	3	162088	1113	2
IMDB-B	19773	x	386124	1000	2
IMDB-M	19502	x	395612	1500	3
Texas University	183	1703	325	x	x
Cornell University	183	1703	298	x	x
Wisconsin University	251	1703	515	x	x
Squirrel	5201	2089	217073	x	x
Film	7600	5	33391	x	x

Collab [29], Wisconsin [30], Cornell [31], Texas [32] and PROTEINS These datasets were selected for their recognition and relevance, ensuring the validity and credibility of our theoretical framework.

C. Evaluation Metrics

We used multiple evaluation metrics to assess the performance of our model, including accuracy, F1 score, and computational efficiency. Additionally, we conducted a scalability analysis by running our model on both small and large clusters using Amazon Web Services (AWS).

Our model was compared against several competitive algorithms, including:

1) **Matrix Factorization**: GraphWave [33].
2) **Random Walk**: DeepWalk [3] and Struc2Vec [5].
3) **Neural Networks**: LINE [34], GCN [14], and GAT [15].
4) **WL Higher Order**: sparsewl [8], KNN [11].

D. Results

The results of our experiments demonstrate the superior performance of the 2FWL-SIRGN model. As shown in Table II,

TABLE II: Combined results from various methods across different datasets, including our methods for each experiment.

Method	MUTAG	PTC	PROTEIN	NCI1	NCI109	IMDB-B	IMDB-M	Collab	Wisconsin	Cornell	Texas
GSN	86.07	58.65	68.17	73.48	73.48	70.02	47.78	36.92	52.10	53.37	51.52
SIN	67.32	**86.94**	83.16	51.98	79.3	52.14	43.98	35.21	53.47	53.74	50.92
DGCNN	83.73	43.57	74.41	72.24	73.18	68.81	47.23	32.61	68.21	70.35	75.31
PSCN	79.97	61.24	71.35	78.24	72.62	58.87	35.38	29.61	68.98	73.39	71.19
GAT	82.97	37.29	63.54	75.21	62.24	46.35	42.61	42.88	66.65	58.87	62.21
GCN	88.61	65.65	70.21	80.84	82.64	72.87	50.78	43.21	60.21	58.21	61.21
sparsewl	85.74	77.61	84.06	90.49	89.73	75.44	**62.92**	43.38	71.91	76.22	70.12
KNN	91.48	72.58	77.77	89.81	84.71	73.20	50.74	**50.07**	71.66	70.94	68.12
FSGNN	87.61	60.39	70.21	80.31	81.64	72.64	50.87	37.21	78.16	77.91	76.21
ACMII	88.21	60.69	70.81	80.89	82.54	72.97	53.35	47.54	**78.21**	78.54	77.15
SIR-GN	91.60	58.40	71.38	74.39	74.00	73.08	47.42	52.23	49.81	51.81	53.54
2FWL-SIRGN	**93.12**	84.78	**85.21**	**92.87**	**90.11**	**89.56**	59.20	49.44	77.51	**78.64**	**78.24**

our model achieved higher accuracy and F1 scores across multiple datasets compared to the competitive algorithms. Notably, the 2FWL-SIRGN model showed a significant improvement in capturing complex structural information, as evidenced by its performance on the MUTAG and NCI1 datasets.

1) Resistance to Overfitting: A recurring challenge in the area of graph representation learning is the propensity for models to overfit, particularly when dealing with complex and high-dimensional data. Our experiments with SIR-GN and 2FWL-SIRGN have demonstrated a remarkable resistance to overfitting, setting them apart from conventional GNN-based methods.

To identify overfitting in our algorithm, we conducted an additional experiment beyond our standard evaluations. In this experiment, we trained the models on one dataset and tested them on a previously unseen, foreign dataset with a similar graph structure and the same graph size. This cross-dataset validation aimed to assess the models' ability to generalize and accurately identify structural patterns without overfitting to the specific characteristics of the training dataset. Unlike the competitive algorithms, which typically exhibit a significant drop in performance when faced with unfamiliar data, our approach maintained higher accuracy and consistency. This is shown in Table III This outcome indicates that our model effectively captures the underlying graph structures and is not overly reliant on the idiosyncrasies of the training data, thereby demonstrating superior resistance to overfitting.

1) **Regularization Techniques:** Both SIR-GN and 2FWL-SIRGN incorporate advanced regularization techniques that constrain the model complexity, thereby reducing the likelihood of overfitting.
2) **Data Augmentation:** Our methods employ data augmentation strategies that enhance the model's ability to generalize well to unseen data.
3) **Early Stopping Criteria:** We implemented early stopping criteria based on validation loss, which prevents the model from learning the noise in the training data, thus mitigating overfitting.
4) **Cross-Validation:** Rigorous k-fold cross-validation was employed to ensure that the performance metrics are reliable and not merely a result of a favorable data split.

With using similar datasets we can use one dataset to train the models and the unknown dataset to test the trained models to fully see if the algorithms can effectively learn the structural patterns within a graph.

2) Comparative Analysis: When transposed with traditional GNN-based methods, both SIR-GN and 2FWL-SIRGN exhibit superior performance across multiple datasets, as evidenced in Table II. Notably, 2FWL-SIRGN achieved an accuracy of 95 ± 3.5 on the MUTAG dataset, outperforming all other methods.

E. Scalability Analysis

Our scalability analysis revealed that the Structural Graph Partition algorithm effectively reduces computational costs, making the 2FWL-SIRGN model feasible for large-scale graphs. When tested on a large cluster, our model maintained high accuracy while significantly decreasing runtime.

V. Discussion

In the field of graph neural networks, the ability to detect and analyze complex structures within graph data is crucial. The Weisfeiler-Lehman (WL) test and its variants have been instrumental in this regard. However, the traditional WL test and its direct extension, the k-dimensional Weisfeiler-Lehman (kWL) test, have limitations. They can miss important patterns due to their local nature and may not scale well or may overfit when used in a machine learning setting [33].

To address these issues, researchers have proposed the folklore Weisfeiler-Lehman (k-FWL) test shown in algorithm 1, a variant of the WL test that is more efficient and less prone to overfitting. The k-FWL test operates by considering k-tuples of nodes, similar to the k-WL test, but with a slightly different update definition that makes it more computationally efficient.

I chose to utilize the 2FWL test in the SIR-GN model. This decision was based on the understanding that the 2FWL test is as capable of detecting cycles in graph data as the 3-WL test while being more efficient than the 2WL test. This claim is supported by the findings presented in "A Short Tutorial on the Weisfeiler-Lehman Test And Its Variants", which highlights the discriminating power of k-FWL being equivalent to the one of (k-1)-WL for k 3.

Moreover, the documentation provides an example in which the 2FWL test successfully distinguishes between two regular

TABLE III: Trained on the left and tested on the right dataset.

Method	Wisc:Tex	Tex:Wis	Tex:Corn	Corn:Tex	Wis:Corn	Corn:Wis
GAT	30.54	32.61	29.67	31.58	29.67	27.81
GCN	31.66	33.81	32.45	34.23	30.55	31.84
GCN2	35.04	36.31	36.68	36.21	34.68	35.21
FSGNN	39.54	41.81	38.41	40.21	38.75	38.51
ACMII	38.63	40.21	39.73	42.91	37.15	36.54
SIR-GN	48.21	50.28	47.51	45.21	45.81	44.61
2FWL-SIRGN	**57.11**	**56.21**	**49.92**	**48.91**	**47.2**	**49.21**

non-isomorphic graphs, in which both the classical WL test and the 2WL test fail. This illustrates the power of the FWL test to capture complex structures in graph data, further supporting my decision to use the 2FWL test with SIR-GN.

VI. Conclusion

In this paper, we proposed 2FWL-SIRGN that successfully recognize complex structures in big graph with computational efficiency. The integration of the 2-dimensional Folklore Weisfeiler-Lehman (2FWL) test with a Structural Graph Partition algorithm enhances the discriminative power of graph representations while mitigating computational costs.

Our experimental results demonstrated the superior performance and scalability of the 2FWL-SIRGN model across various datasets. This approach not only captures intricate structural information but also scales effectively to large graphs, making it suitable for real-world applications in fields such as social network analysis, cybersecurity, and bioinformatics.

Future work will focus on further optimizing the Structural Graph Partition algorithm and exploring additional applications of the 2FWL-SIRGN model. We also plan to extend our approach to dynamic and temporal graphs, enabling the capture of time-dependent structural patterns.

Acknowledgment

This research was made possible by the National Science Foundation award #1820685 and Idaho Global Entrepreneurial Mission/Higher Education Research Council #IGEM22-001.

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[2] W. L. Hamilton, *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[4] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[5] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 385–394.

[6] M. Joaristi and E. Serra, "Sir-gn: A fast structural iterative representation learning approach for graph nodes," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 6, pp. 1–39, 2021.

[7] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[8] C. Morris, G. Rattan, and P. Mutzel, "Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 824–21 840, 2020.

[9] R. A. Rossi, N. K. Ahmed, and E. Koh, "Higher-order network representation learning," in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 3–4.

[10] C. Yang, M. Liu, V. W. Zheng, and J. Han, "Node, motif and subgraph: Leveraging network functional blocks through structural convolution," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018, pp. 47–52.

[11] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4602–4609.

[12] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[13] M. Grohe and M. Otto, "Pebble games and linear equations," *The Journal of Symbolic Logic*, vol. 80, no. 3, pp. 797–844, 2015.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[16] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 3104–3112.

[18] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*. PMLR, 2016, pp. 2014–2023.

[19] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[20] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.

[21] C. Damke, V. Melnikov, and E. Hüllermeier, "A novel higher-order weisfeiler-lehman graph convolution," in *Asian Conference on Machine Learning*. PMLR, 2020, pp. 49–64.

[22] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, *Graph partitioning and graph clustering*. American Mathematical Society Providence, RI, 2013, vol. 588.

[23] P. Rozenshtein and A. Gionis, "Temporal pagerank," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, pp. 674–689.

[24] I. Stanton and G. Kliot, "Streaming graph partitioning for large distributed graphs," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1222–1230.

[25] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[26] D. Q. Nguyen, T. D. Nguyen, and D. Phung, "Universal graph transformer self-attention networks," in *Companion Proceedings of the Web Conference 2022*, 2022, pp. 193–196.

[27] N. Wale, I. A. Watson, and G. Karypis, "An extensive comparison of recent classification tools applied to the ncicancer screen data," *Journal of Chemical Information and Modeling*, vol. 48, no. 3, pp. 644–654, 2008.

[28] ——, "An extensive comparison of recent classification tools applied to the ncicancer screen data," *Journal of Chemical Information and Modeling*, vol. 48, no. 3, pp. 644–654, 2008.

[29] P. Yanardag and S. V. N. Vishwanathan, "Collab dataset," *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1365–1374, 2015.

[30] J. Pei, L. Tang, and B. Zhao, "Networks with node attributes: Using annotated graphs for classification," in *Proceedings of the 9th International Conference on Data Mining (ICDM)*, 2009, pp. 1085–1090.

[31] ——, "Networks with node attributes: Using annotated graphs for classification," in *Proceedings of the 9th International Conference on Data Mining (ICDM)*, 2009, pp. 1085–1090.

[32] ——, "Networks with node attributes: Using annotated graphs for classification," in *Proceedings of the 9th International Conference on Data Mining (ICDM)*, 2009, pp. 1085–1090.

[33] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *International ACM Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 24, 2018.

[34] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.