



Graph neural networks for positive and unlabeled learning: a rewiring approach

Guilherme Henrique Messias¹ · Sylvia Iasulaitis² · Alan Demétrius Baria Valejo¹

Received: 23 January 2025 / Revised: 8 July 2025 / Accepted: 12 July 2025 /

Published online: 1 August 2025

© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2025

Abstract

Graph rewiring strategies have emerged as powerful tools to enhance vertex connection within a graph. These connections can be used by graph neural networks (GNNs) to improve classifier performance. However, rewiring methods are unsupervised and aim to improve connection metrics. This work introduces the positive sequential rewiring via breadth search (PSRB) model, an approach that applies supervised rewiring to the available classes in a positive and unlabeled learning (PUL) problem. PSRB employs a probability density function to select edge addition between positive vertices, facilitating the sequential application of a relational GNN. Experiments on five datasets from the literature show the algorithm's efficiency compared to state-of-the-art two-step PUL algorithms.

Keywords Positive and unlabeled learning · Graph neural networks · Rewiring

1 Introduction

Profound technological transformations in the information society have led to a massive volume of data being produced and accessed at an impressive speed. Data capture and processing have become highly strategic. The advancement of machine learning algorithms has enabled the execution of complex tasks, such as classifying vast amounts of data.

For simpler classification tasks, binary classification algorithms can offer a viable solution. These algorithms rely on training data where both class labels are available to the classifier. However, in real-world scenarios, labeling large datasets can be challenging.

A research approach for extracting information from vast databases is positive and unlabeled learning (PUL). In PUL, learning occurs only with positive (class of interest) and

✉ Guilherme Henrique Messias
ghmessias@estudante.ufscar.br

Sylvia Iasulaitis
si@ufscar.br

Alan Demétrius Baria Valejo
alanvalejo@ufscar.br

¹ Departamento de Computação, UFSCar, Rod. Washington Luís, São Carlos, SP, Brazil

² Departamento de Ciências Sociais, UFSCar, Rod. Washington Luís, São Carlos, SP, Brazil

unlabeled data. This approach is a variant of the binary learning concept, where the classifier is trained by having access solely to unlabeled examples and to labels of positive examples [1].

PUL algorithms can use various classification methods, such as (1) biased learning, (2) class-prior incorporation and (3) two-step techniques. Biased learning algorithms treat unlabeled data as negative examples with noise, utilizing methods such as expectation maximization [2], bagging, support vector machines (SVMs) [3, 4] and Laplacian learning on hyperplanes [5]. The class-prior incorporation method uses the probability distribution of label frequencies, applied through weighted logistic regression [6] or by directly incorporating label probabilities [7].

The two-step technique, the focus of this article, involves identifying some reliable negative examples before classifying the remaining examples. Reliable negative examples can be found using SVMs [8] or clustering algorithms such as agglomerative clustering [9] or *k*-means [10]. For PUL in graphs, reliable negative examples can be detected using global similarity matrices [11] or Dijkstra's algorithm [12].

In the field of graph learning, graph neural networks (GNNs) have shown promising results in areas such as network traffic identification [13, 14], edge prediction in social networks [15], network modeling [16] and text classification [17]. While GNNs have been widely applied across various domains, few studies considered their use for classifying PUL problems. Wu et al. [18] pioneered applying GNNs–PUL, using the class-prior incorporation method.

However, to the best of our knowledge, no studies have investigated the use of GNNs for PUL using the two-step technique. To address this gap, we propose two novel techniques in this study:

1. To improve vertex connectivity and locality, we propose the PSRB model, which sequentially rewires positive vertices to facilitate message passing between them, generating new graphs with improved connections. These new graphs are then used to propagate information between positive vertices in a GNN. To deal with these different graphs, we propose adapting the relational graph convolutional network (RGCN) model [19]. Different from vanilla graph convolutional network (GCN) [20], RGCN uses a relational information to aggregate features from different sets of neighbors, in our case, different graphs. After learning the latent representation, the similarity is used to label reliable negative data.
2. In the second step, a multilayer perceptron (MLP) model is employed to classify the remaining examples. Other binary classification methods, such as SVMs, can also be used. The binary classifier is trained using both the positive and reliable negative labels.

Experiments conducted on real-world datasets demonstrate the effectiveness of the proposed algorithm, particularly in the inference of reliable negative data. The experiments were carried out in three phases: (1) a parametric analysis of the PSRB model to determine the optimal values for rewiring; (2) a benchmark comparison of the proposed model against five classical and contemporary algorithms from the literature (based on the two-step technique) for reliable negative data inference; and (3) an evaluation of the classification of all dataset examples, comparing the performance of our model with existing algorithms.

Thus, this work contributes in several ways: introducing the PSRB rewiring model for PUL tasks, proposing a relational model to handle sequentially generated graphs, conducting a parametric analysis of the PSRB model and providing an experimental comparison with literature-based algorithms. Additionally, we use the GCN model [20] as a baseline for learning the latent representation and inferring reliable negative data.

This article is divided as follows: Sect. 2 presents the basic concepts used in the research. Section 3 presents the proposed approach. Section 4 details the parametric tests and experiments of the first and second stages of the algorithm. Finally, Sect. 5 presents the conclusion.

2 Background and related works

This section presents the basic concepts used in the research: graphs, GNNs, autoencoders, rewiring strategies and PUL.

2.1 Graphs

A graph $G = (V, E, X, Y)$ is a mathematical structure composed of a set of vertices V of size n , a set of edges $E = \{(u, v) : u, v \in V\}$, a matrix $X_{b \times n}$ where $x_u \in \mathbb{R}^b$ are the b -dimensional feature vectors of the vertices u and $Y = \{y_i | i = 1, \dots, n\}$ is the set of labels of each vertex. The vertices of a graph can be represented by a matrix $A_{n \times n}$ where $A_{i,j} = 1$ if $(i, j) \in E$ and 0 otherwise [21].

2.2 Graph neural networks

GNNs are neural network architectures that consider information from feature matrix X and the graph topology [22]. These architectures have been used in different data domains, such as computer vision [23], recommendation systems [24], transportation [25] and drug interactions [26].

Among the various GNN models, message-passing neural networks (MPNNs) stand out for their ability to learn representations of graph structures. They operate by transferring information between vertices. Considering UP as a weight update function and AGG as a weight aggregation function (like the average, for example), an MPNN model consists of stacking T layers:

$$x_v^{(t)} = \text{UP}^{(t)} \left(x_v^{(t-1)}, \text{AGG}^{(t)} \left(\left\{ x_u^{(t-1)} : (u, v) \in E \right\} \right) \right), \quad (1)$$

where $t = 1, \dots, T$.

GCNs are one of several types of MPNNs. Bruna et al. [27] showed the use of convolutional layers in low-scale graphs. Kipf and Welling [20] formalized the concept of GCN, proposing an architecture that considers the local neighborhood of a vertex to aggregate information, generating a semi-supervised classification model when dealing with robust graphs. Yuan et al. [28] propose consistent GNNs, a model that uses contrastive learning to learn graphs with noise, using regularization and detecting nodes breaking the homophily assumption as noisy to clear the label noise.

GCNs are functions defined on a feature graph that can have outputs in vertices, edges or complete graphs. The GCN model consists of stacking T vertices as follows:

$$H^{(t+1)} = \sigma \left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(t)} W^{(t)} \right), \quad (2)$$

where $\tilde{D}$ is the degree matrix over the adjacency matrix $\tilde{A} = A + I$, σ is a nonlinear activation function and W is a matrix of trainable weights.

Among several GNN models, the RGCN model [19] is used in this research. This model aggregates relations defined in a graph separated from message passing. Considering a relational feature graph, the update equation of each layer is given by

$$h_i^{(l+1)} = \sigma \left(\sum_{r \in \mathcal{R}} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)} \right) \quad (3)$$

where N_i^r is the set of neighbors of i over the relation $r \in \mathcal{R}$ and $c_{i,r}$ is a normalization constant. In this work, we use the RGCN model due to its ability to handle multiple relations between vertices at different stages. Here, each graph generated by the rewiring algorithm is considered a distinct relation. This approach differs from the traditional GCN by enabling the model to process multiple graphs. Moreover, other models can be employed to compute the normalization constant, such as graph attention networks (GAT) [29] and GraphSAGE [30].

GCNs are recognized for their applicability in large-scale graphs, as the aggregation function operates locally with each vertex using a direct neighborhood [18, 20]. Despite efficient, this aggregation has difficulties to capture information across longer distances. A possible solution to this problem is to stack multiple GCN layers to extend the network's reach to more distant vertices. However, this approach introduces the issue of over-squashing [31], i.e., the network struggles to summarize the amount of information in a finite vector.

Rewiring methods have been proposed to address this problem, reducing the distances between vertices in the graph, by either using a similarity metric or locally densifying vertices.

2.3 Graph rewiring

The main idea of using rewiring in GNNs is to decompose the original graph into graphs used to compute the latent representation. Thus, rewiring strategies consist of applying an operator $R-G = (V, E, X, Y)$, producing a new graph $R(G) = (V, R(E), X, Y)$ with the same vertices but with different connectivity. Then, an MPNN model can aggregate information from a neighborhood in both G and $R(G)$. Considering an MPNN from a rewiring perspective, we define the update in each layer t as:

$$x_v^{(t)} = \text{UP}^{(t)} \left(x_v^{(t-1)}, \text{AGG}_G^{(t)} \left(\left\{ x_u^{(t-1)} : (u, v) \in E \right\} \right), \text{AGG}_{R(G)}^{(t)} \left(\left\{ x_u^{(t-1)} : (u, v) \in R(E) \right\} \right) \right). \quad (4)$$

Several models in the literature consider the function $\text{AGG}_G = 0$, updating the representations only with information from the graph after the rewiring operation. The methods in the literature use two approaches for rewiring strategies: (1) spectral models and (2) spatial models.

Spatial rewiring models focus on localized operations at each vertex, directly aggregating information within the MPNN. Brüel-Gabrielsson et al. [32] use direct connections of neighbors of an r -neighborhood. Abboud et al. [33] use shortest paths to propagate information in an MPNN. Graph transformer architectures are also widely used for rewiring-based solutions [34–36]. These models reduce over-squashing by creating new edges in the original graph, reducing the diameter and increasing information propagation throughout the network.

On the other hand, spectral methods consider global operations on the graph. According to [37], two primary approaches are explored within spectral methods: (1) using the spectral gap [38] and (2) leveraging effective resistance [39]. These methods add a limited number of edges, resulting in the improvement of specific connectivity metrics.

Most rewiring algorithms, especially spectral ones, add edges instantly. (The result consists of only one graph with more edges.) The Locality-Aware SEquential Rewiring (LASER) algorithm [37] uses a sequential approach to create the final graph from a combination of

spectral and spatial approaches. The graph's topology is changed at each step of the algorithm, generating different graphs.

In LASER, for a node v in stage l , the aim is to add a direct edge of vertices at a distance $l + 1$ from v . A parameter ρ is used to add edges with a lower score of connectivity μ . Eq. 5 exemplifies the sequential creation of graphs:

$$G = G_0 \xrightarrow{R_1} G_1 \xrightarrow{R_2} G_2 \cdots \xrightarrow{R_L} G_L. \quad (5)$$

Thus, the sequential creation of edges allows a smoother evolution from graph G_0 , more robust to over-squashing, preserving information about the graph topology [37].

The information originally present only in graph G_0 is now split among the new graphs created sequentially. The GNN model is proposed to consider all these graphs generated at each layer.

2.4 Graph AutoEncoders

Autoencoders are a type of neural network architecture extensively used in deep learning, particularly for handling large datasets. They are essential for reducing computational complexity and mitigating the curse of dimensionality [40].

The autoencoder model applies two transformations to the original data X . The first, the encoder, transforms the original data X into a lower-dimensional space $h(X)$. The second, the decoder, reconstructs the data to its original space. These transformations enable the model to learn the distribution of the data.

Kipf and Welling [41] extended the autoencoder concept to graph data, starting with a graph G and a feature matrix X . The encoder can be implemented using a simple GCN (Eq. 2). The decoder reconstructs the adjacency matrix A by multiplying the transposed matrices, allowing the model to adjust weights to reduce a reconstruction error.

Thus, the reduced-dimensional space captures both the graph's feature matrix X and the connectivity of the vertices from matrix A . Graph autoencoder (GAE) models will be a baseline for comparison with the proposed algorithm. This is the first time this approach has been used for this purpose.

2.5 Positive and unlabeled graph learning

This section presents the PUL-related works that provide the foundation for this study.

2.5.1 Problem definition

The PUL learning paradigm is a special case of binary classification [1]. Formally, given a set of vertices $\mathcal{V} = \{v_1, \dots, v_l, v_{l+1}, \dots, v_{l+u}\} \subset \mathbb{R}^m$ and a class of labels $\mathcal{C} = \{-1, 1\}$, where 1 represents the positive class and -1 represents the negative class, the first l elements of $\mathcal{V}$ are labeled as positive class 1, forming the set of labeled examples $\mathcal{P}$. The remaining u elements of $\mathcal{V}$ are unlabeled examples, forming the set of unlabeled examples $\mathcal{U}$, and $u \gg l$ and $|\mathcal{V}| = |\mathcal{P} \cup \mathcal{U}|$.

Similarly, a PUL algorithm on graphs seeks to label vertices present in $\mathcal{U}$. Thus, it is considered a PUL algorithm when it learns from the set $\mathcal{P}$ and labels the examples present in $\mathcal{U}$ [11].

2.5.2 Related works in PUL

PUL approaches can be divided into three methods: biased learning, class-prior incorporation and the two-step technique [1]. Algorithms based on the two-step technique, as those examined in this research, consist of (1) finding negative examples from positive data and (2) classifying the remaining data with a binary classifier.

In the first step, there are several strategies, some of which are based on data grouping using k -means (maximum margin clustering with least squares—MCLS) [10] or the use of agglomerative clustering (clustering-based method for collecting reliable negative examples—CCRNE) [9], while others use the support vector machine (Rocchio support vector machine—RCSVM) [8]. For graph-based algorithms, negative data can be inferred through the distance between vertices in a graph (label propagation for PUL—LP_PUL) [12] or with the use of centrality matrices (PUL via label propagation—PU_LP) [11].

For the second step, most algorithms use SVMs or label propagation algorithms, such as learning with local and global consistency (LLGC) [42].

Wu et al. [18] pioneered using GNNs for learning in PUL problems, using class-prior incorporation and considering each vertex's global and local neighborhood. To our knowledge, no PUL models have been found that use GNN based on the two-step technique.

3 Proposed approach

This section presents the proposed approach for a PUL model based on the two-step strategy.

The proposed model begins with a rewiring strategy directly applied to the positive data aiming to improve connectivity and, consequently, message passing between these vertices. This initial step employs a RGCN model to learn the latent representation of the graph. To infer the reliable negative examples, each unlabeled example is ranked according to the similarity of the positive examples, and those with the lowest similarity are chosen. For the second step, a binary classifier is used to classify the remaining examples.

3.1 Negative inference

The model applies two strategies to perform the first step and infer the reliable negative data: first, a rewiring model capable of adding edges between positive vertices is applied. Then, a relational GCN model is used to learn a latent representation that will be used to infer the reliable negative data.

3.1.1 Rewiring

Stacking GCN layers to aggregate information over a larger radius is inefficient in most datasets [20]. Thus, rewiring strategies prove to be consistent in mitigating this problem [43].

The LASER algorithm, proposed by Barbeto et al. [37], connects a portion ρ of vertices that improve a connectivity measure μ of the graph. The connectivity measure μ may be different for each problem, making it challenging to formulate an ideal measure.

To develop a rewiring model that can more effectively adapt to various PUL problems, this work proposes a sequential algorithm that employs a probability density function to generate new graphs by selectively connecting positive vertices. The rewiring strategy is restricted to the set $\bar{E} = \{(u, v) : u, v \in \mathcal{P}\}$.

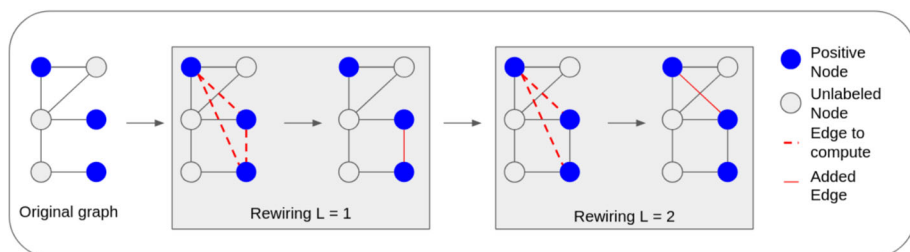


Fig. 1 Illustration of the proposed rewiring approach. The rewiring is applied for each value of L , resulting in a different graph used by the model to aggregate information from different neighborhoods

The probability density function is modeled based on the following key premises:

- The algorithm should prioritize enhancing communication between distant positive vertices to minimize the need for additional layers. Additionally, it must allow for the connection of vertices across disconnected components.
- The algorithm must account for the number of positive vertices, as a high volume of positive vertices can increase the probability of adding edges and thereby densify the graph.
- During each iteration, the algorithm should aim to add fewer edges to preserve the graph's sparseness.

Therefore, the probability density function ϕ is defined as:

$$\phi(u, v|P, L) = \frac{1}{d(u, v)^\alpha \cdot |P|^\beta \cdot L^\gamma}, \quad (6)$$

where $|P|$ represents the number of positive labeled elements and $\alpha, \beta, \gamma \in \mathbb{R}$. The distance $d(u, v)$ is computed using breadth-first search (BFS), which is particularly well suited for unweighted graphs. We chose BFS because it efficiently identifies the shortest topological distance between nodes, offering reliable nearest-neighbor information without the computational complexity associated with more elaborate distance heuristics designed for weighted graphs.

Thus, the algorithm first calculates the probability of adding each edge (u, v) according to Eq. 6, after which a value is generated from a uniform distribution to define whether the edge will be added. A new graph is formed at the end of calculating all edges. The process can be repeated L times. The choice of this probability density function is justified by:

1. To preserve locality, the distance must be considered. Also, the possibility of connecting vertices with infinite distance (vertices of distinct connected components) by choosing $\alpha = 0$ must be guaranteed.
2. The number of positive elements must be considered when adding edges. However, the graph must be sparse since very large datasets can result in a very large addition of edges.
3. The number of graphs generated must be considered when choosing whether to add new edges, to have an upper bound for the number of elements in E .

Figure 1 shows the application of the rewiring strategy proposed by Algorithm 1.

3.1.2 Graph neural network model

We adapted the relational GCN proposed by Babeto et al. [37] and Karhadkar et al. [44] to deal with L graphs generated from the rewiring model. Considering a GCN (described in

Algorithm 1 Positive sequential rewiring via breadth search

Require: Set of positive vertices P , graph G , probability function parameters α, β, γ , number of iterations L .

Ensure: Rewired graphs *rewiring_graphs*.

```

1: rewiring_graphs  $\leftarrow []$ 
2:  $G' \leftarrow G$ 
3: for  $l \in [1, L]$  do
4:   for all  $(u, v) \in P$  do
5:     Calculate BFS distance  $d(u, v)$  between  $u$  and  $v$  in  $G'$ 
6:     Calculate probability  $\phi(u, v|P, L) \leftarrow \frac{1}{d(u, v)^\alpha \cdot |P|^\beta \cdot L^\gamma}$ 
7:     Generate random number  $r \sim \text{UNIFORM}(0, 1)$ 
8:     if  $r < \phi(u, v|P, L)$  then
9:       Add edge  $(u, v)$  to  $G'$ 
10:    end if
11:  end for
12:  Add  $G'$  to rewiring_graphs
13: end for
14: return rewiring_graphs

```

Eq. 2) and based on the rewiring perspective of Eq. 5, the proposed model consists of stacking T layers based on Eqs. 7–9:

$$S_0^{(t)}(v) = \sum_{u \in N(v) \cup v} \frac{1}{\sqrt{d_u d_v}} W_0^{(t)} x_u^{(t-1)} \quad (7)$$

$$S_L^{(t)}(v) = \sum_{l=1}^L \sum_{u \in N_\phi^l(v) \cup \{v\}} \frac{1}{\sqrt{d_{u,l} d_{v,l}}} W_l^{(t)} x_u^{(t-1)} \quad (8)$$

$$x_v^{(t)} = \sigma \left(S_0^{(t)}(v) + S_L^{(t)}(v) \right) \quad (9)$$

where d_u is the degree of vertex u , $d_{u,l}$ is the degree of u in graph l and $N_\phi^l(v)$ is the set of neighbors of v from the l th iteration of the algorithm based on the probability density function ϕ .

The model defined in Eq. 9 will function as the encoder in a GAE framework, following the approach established by Kipf and Welling [41]. The decoder will be implemented by multiplying the encoder's result by its transpose. Thus, the training process yields an embedding that considers the density of connections generated by rewiring and the graph's feature matrix.

For negative data inference, we employ a similarity measure within the latent representation space produced by the embedding, classifying the *num_neg* most distant examples as negative. In this study, we use Euclidean distance for this purpose. We justify the use of Euclidean distance based on the nature of the problem, considering that positive vertices will increase the density of points around them, making Euclidean distance the most appropriate way to find less similar vertices. Algorithm 2 presents the procedure for reliable negative data inference.

3.2 Binary classifier

After labeling the negative data with the proposed model, the second step of the algorithm involves using a binary classifier to classify the remaining examples. This classifier will

Algorithm 2 Negative extraction via rewiring GAE

Require: A trained GAE model and graph G , P is the positive set, U is the unlabeled set, num_neg is the number of negatives to extract.

Ensure: A labeled graph with negatives labeled in U .

```

1:  $model \leftarrow \text{Train}(G)$ 
2:  $lat\_rep \leftarrow \text{GAE\_Encode}(model, G)$ 
3:  $avg\_distances \leftarrow \{\}$ 
4: for  $u \in U$  do
5:    $dist \leftarrow 0$ 
6:   for  $v \in P$  do
7:      $dist \leftarrow dist + \text{Distance}(lat\_rep[u], lat\_rep[v])$ 
8:   end for
9:    $avg\_distances[u] \leftarrow dist/|P|$ 
10: end for
11:  $sorted\_vertices \leftarrow \text{Sort\_By\_Avg\_Dist}(avg\_distances)$ 
12: for  $i \leftarrow 1$  to  $num\_neg$  do
13:    $u \leftarrow sorted\_vertices[i]$ 
14:    $\text{Label}(u) \leftarrow \text{"negative"}$ 
15: end for
16: return  $\text{Labeled\_Graph}(G, P, U)$ 

```

distinguish between the positive data and the reliable negative data identified in the first step. Any algorithms can be employed for this purpose, including SVMs, neural networks or label propagation algorithms. In this study, we used a MLP neural network for classification.

4 Experiments

4.1 Datasets and baselines

Three distinct experiments were conducted to evaluate the proposed algorithm. First, we tested the algorithm with various parameter settings to assess the impact of each parameter. Next, we compared the algorithm against five established methods from the literature to determine its effectiveness in inferring reliable negative data from positive examples. Finally, we evaluated the general classification performance by comparing the same five algorithms in terms of their $F1$ scores across all examples.

The PU_LP and LP_PUL algorithms employed the LLGC method to infer the labels of all examples. The GAE algorithm used the GCN architecture, while the remaining algorithms used traditional SVMs via the scikit-learn library [45].

The datasets used in the experiments were Cora, CiteSeer, PubMed, Amazon Photo and DBLP [46–48], with positive class labels of 3, 2, 2, 6 and 0, respectively. These class selections correspond to the most frequent elements in each dataset, effectively converting the multiclass classification problems into binary classification tasks, as described in [18].

Table 1 provides detailed statistics for the selected datasets, including the number of vertices, edges, graph density, percentage and absolute count of positive vertices, and the number of disconnected components. Density is defined as the ratio of the number of edges to the square of the number of vertices.

To ensure a fair comparison, the proposed model was tested against the following baseline algorithms with necessary adjustments, fixing the number of reliable negative examples to match the number of positive data:

Table 1 Statistics of datasets

Dataset	Nodes	Edges	Density	Features	Positives (%)	Positives (abs)	# of components
Cora	2708	5278	0.0007	1433	30.21	818	78
CiteSeer	3327	4552	0.0004	3703	20.08	668	438
PubMed	19,717	44,302	0.0001	500	39.09	7875	1
Photo	7650	119,081	0.0002	745	25.37	1941	136
DBLP	17,716	52,867	0.0003	1639	44.71	7920	589

- **MCLS**: For k -means clustering, a value of $k = 7$ was used. To prevent the set of positive clusters from being empty, clusters obtaining a higher average of positives were classified as positive.
- **CCRNE**: To prevent the application of the algorithm from being unable to return reliable negative elements, a ratio of $r = 0.3$ was considered, reducing the distance from the positive centroids and preventing the set of negatives from being empty.
- **RCSVM**: To apply this algorithm, the Euclidean distance and the values of $\alpha = 0.7$, $\beta = 0.3$.
- **PU_LP**: The parameters used to execute this algorithm were $\alpha = 0.1$, $\lambda = 1$, $m = 3$.
- **LP_PUL**: The LP_PUL algorithm uses the distance through Dijkstra's algorithm to find the most distant elements without the need for additional parameters
- **GCN**: The parameters used were $N_LAYERS = [NUM_FEATURES, 64, 16]$ with Adam optimizer, using the PyTorch geometric library [49]. To classify the negative data, a new GCN layer trained with the positive data and negative data obtained in the first step was coupled.
- **PSRB**: For the proposed approach, values of $L = 2$, $N_LAYERS = [NUM_FEATURES, 64, 16]$ was used. To label the remaining examples, an MLP layer was used at the end of the GAE embedding, coupled with the last RGCN layer. We also evaluated PSRB using relational GAT and relational GraphSAGE models, referred to as PSRB (RGAT) and PSRB (RGraphSAGE), respectively. The original proposed approach with RGCN is referred as PSRB (RGCN).

The experiments are available here <https://github.com/GHMessias/psrb>.

4.2 Parametric evaluation

Two separate experiments were carried out to evaluate the parameters of the proposed model. The first sought to understand the influences of the probability parameters α , β e γ on rewiring. The second sought to find the best value of L for each dataset, showing how the algorithm behaved with information from a larger number of graphs.

4.2.1 Probability values

To understand the influence of the probability parameters α , β e γ on rewiring, tests were performed with the values of $\alpha = [0, 0.1, 0.2, 0.3, 0.4]$, $\beta = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ and $\gamma = [0, 1, 2, 3, 4]$ The tests adopted the parameters $L = 2$, and the rate of positive data was fixed at 0.1. Table 2 illustrates the average of the 10 highest and lowest $F1$ score values obtained for different parameters tested.

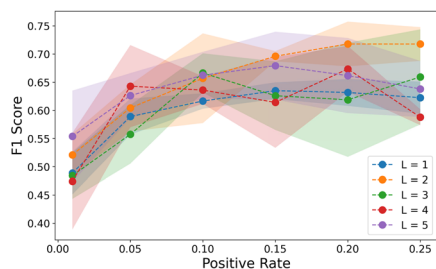
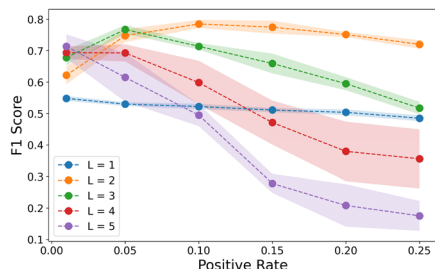
The α values consider the distance between two vertices for adding a new edge (Eq. 6). For sets with a large number of connected components, the value of $\alpha = 0$ implies that there will be a connection between connected components.

The parameter β consists of the variation of the probability of adding new edges based on the amount of positive data. The results suggest that this value is the most important in the network. For datasets with a large number of positive vertices and high density, a higher β value is considered compared to datasets with a large number of positive vertices and low density. For datasets with fewer positive vertices and medium density, β values close to 0.5 are considered.

The γ parameter works to contain the excessive addition of edges as the number of graphs generated by rewiring increases. We recommend using a large α for graphs with smaller

Table 2 Highest and lowest scores according to parameters α , β and γ

Dataset	Highest scores				Lowest scores			
	α	β	γ	F1 score	α	β	γ	F1 score
Amazon photo	0.15	0.11	1.6	0.6438	0.09	0.63	2.3	0.4103
DBLP	0.2	0.71	1.7	0.8613	0.14	0.04	2.8	0.4301
CiteSeer	0	0.56	2	0.5743	0.2	0.27	3.4	0.2989
Cora	0.23	0.46	1.6	0.7447	0	0.15	1.6	0.3468
PubMed	0.17	0.36	2.2	0.8049	0.29	0.93	2.8	0.5628

(a) F1 score for Cora dataset for different values of L (b) F1 score for PubMed dataset for different values of L **Fig. 2** Comparison of F1 scores for different datasets and values of L

radius, smaller β values for large training sets and setting γ proportionally to the number L of graphs generated by the rewiring model.

4.2.2 Number of generated graphs

To evaluate the algorithm's efficiency based on the L parameter, 5 samples for values of 0.01, 0.05, 0.10, 0.15, 0.20 and 0.25 of labeled data for the Cora and PubMed datasets are considered. To calculate the rewiring probability, we used values of $\alpha = 0.23$, $\beta = 0.46$, $\gamma = 2$ for the Cora dataset and $\alpha = 0.17$, $\beta = 0.36$, $\gamma = 2$ for the PubMed dataset. Figure 2a, b illustrates the results obtained from the experiments.

The algorithm performs differently for different values of L . As the L values increase, the algorithm's score decreases. This phenomenon is explained by over-smoothing since many L values cause the latent representation of a vertex to aggregate a lot of information in a vector of fixed size. This occurrence is similar to the experiments carried out by Kipf and Welling [20] and by Wu et al. [18], where many convolution layers or the aggregation of very distinct neighborhoods resulted in low scores. In both experiments, the value of $L = 2$ presented the most relevant results.

4.2.3 Epochs of training

Given the number of generated graphs in the RGCN model, PSRB may be affected by over-fitting. To investigate this behavior within the proposed method and to provide more tailored recommendations for specific use cases, we conducted two experiments focusing on the

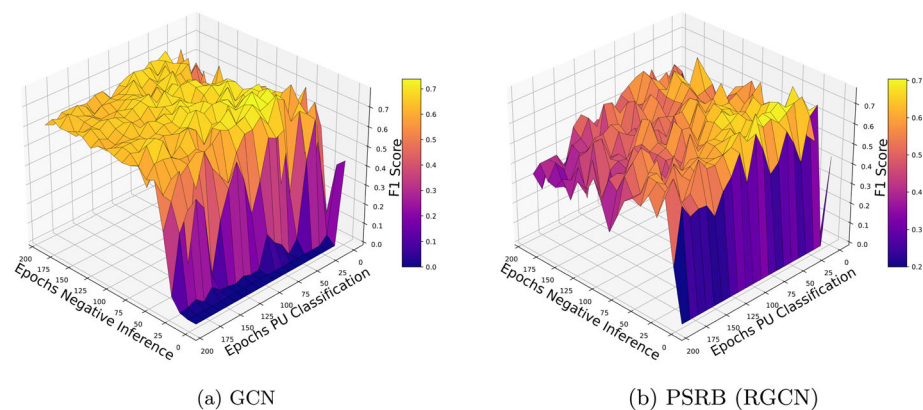


Fig. 3 *F1* score per epoch for the negative data inference and binary classification models

number of training epochs. In both cases, we fixed the number of positive samples ($p = 0.1$) and varied the number of epochs across three different sample sets. Figure 3 illustrates the algorithm's behavior as a function of the number of epochs in the binary classification task and negative inference task.

It is evident that, in the PUL scenario, GNN-based models tend to achieve optimal performance with a relatively small number of training epochs: around 30 epochs for the negative inference model (GCN or PSRB) and approximately 50 epochs for the binary classifier. Based on this observation, these values were adopted for the Cora, CiteSeer and Amazon Photo datasets to perform negative data inference for the binary classification phase. For the remaining datasets, 100 epochs were used for each classifier, considering their higher structural complexity in terms of features, edges and nodes.

4.3 Experiments on negative inference

Positive data values, labeled within a unitary interval of 1–25%, were used to assess the quality of negative example extraction. The negative values to be inferred correspond to the same range as the positive values. Each algorithm was trained five times on each dataset (Cora, CiteSeer and PubMed), and the reported results represent the average performance across these training iterations. To evaluate the quality of the embeddings generated by the model, we conducted an additional benchmark against DropEdge [50] and SkipNode [51], both applied within a GCN framework. This comparative analysis aims to provide further evidence that PSRB can also mitigate the over-smoothing issue in certain scenarios.

Figure 4 shows the results of the negative data inference experiments on the CiteSeer dataset. The LP_PUL algorithm achieved the best performance, demonstrating strong capability in inferring negative examples. The proposed approach, PSRB, also yielded competitive results. In contrast, the RCSVM algorithm failed to infer negative data effectively on this dataset. The SkipNode algorithm showed limited performance, while DropEdge, despite achieving good results, did not outperform the proposed method.

For the Cora dataset (Fig. 5), the PU_LP algorithm achieved the highest overall *F1* score metrics. However, the proposed approach also proved competitive, showing greater generalization and stability throughout the experiments. The SkipNode algorithm exhibited a higher

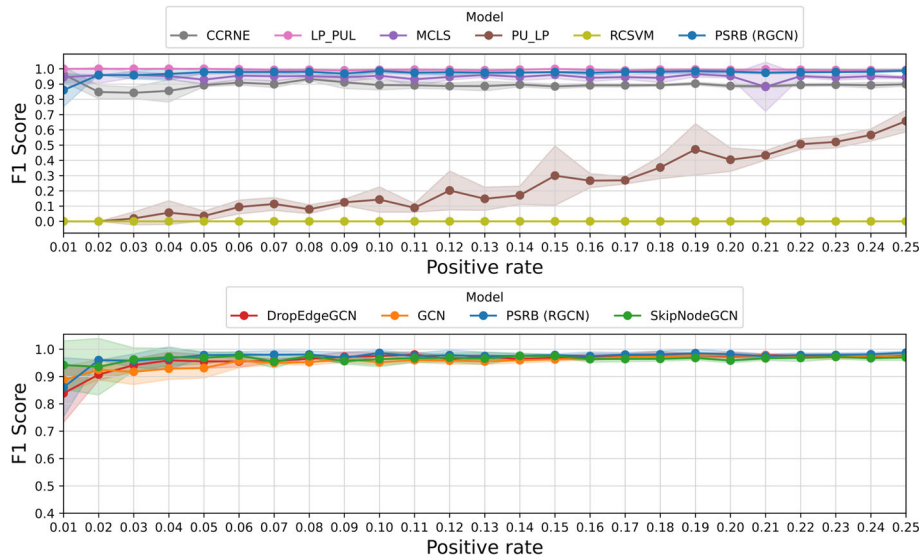


Fig. 4 Comparison of $F1$ scores for the CiteSeer dataset under different parameter settings

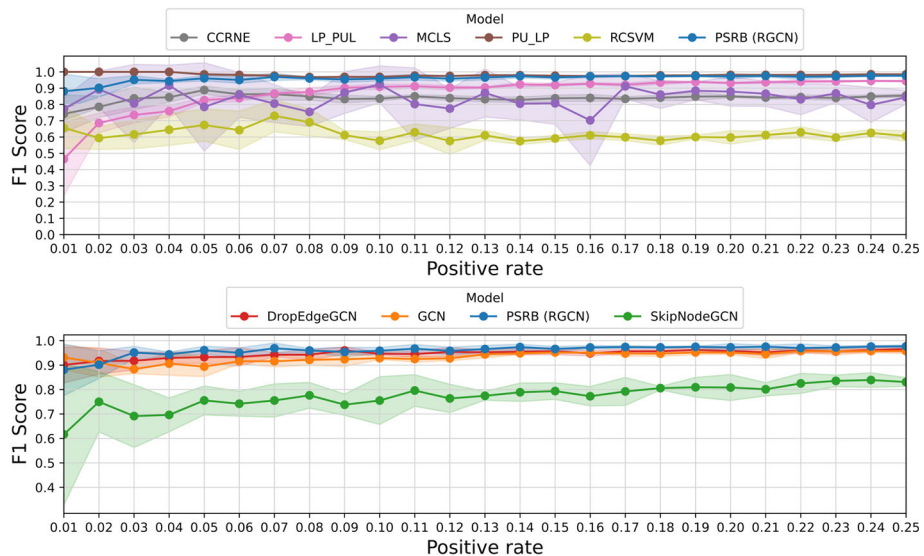


Fig. 5 Comparison of $F1$ scores for the Cora dataset under different parameter settings

standard deviation and greater instability for lower values of p , whereas both DropEdge and PSRB consistently delivered the best performance.

On the PubMed dataset (Fig. 6), the proposed PSRB algorithm outperformed all other algorithms from the literature, particularly for values of $p > 0.7$.

In all experiments, the PSRB algorithm achieved performance that was comparable to or better than that of existing methods, demonstrating its effectiveness in extracting negative

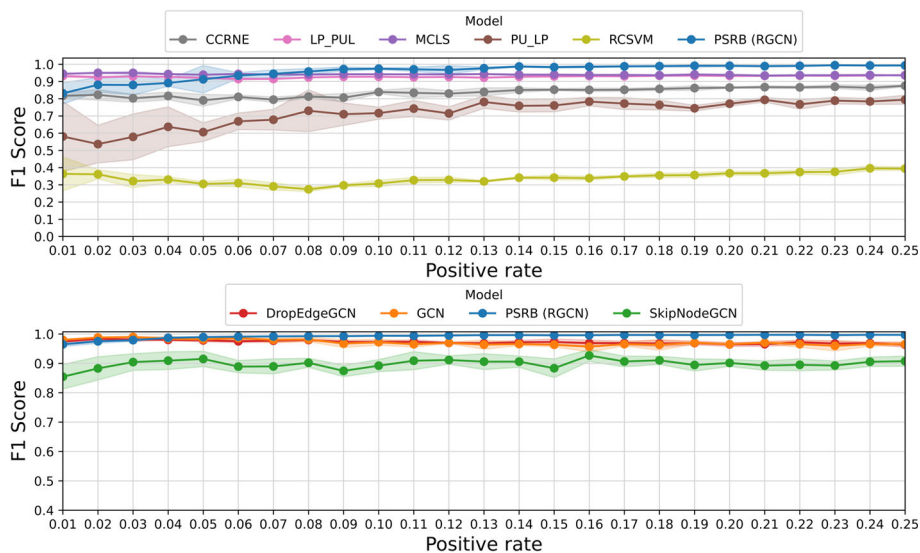


Fig. 6 Comparison of $F1$ scores for the PubMed dataset under different values of parameter p

examples from positive data. This strong performance is reflected in the binary classification task, as discussed in the following chapter.

4.4 Experiments on positive and unlabeled classification

The same datasets were used to evaluate the classification of the examples. The algorithms were tested to infer equal negative and positive elements. Tables 3, 4, 5, 6 and 7 present the results for the Photo, DBLP, CiteSeer, Cora and PubMed datasets.

The analysis of the results for the Amazon Photo dataset (Table 3) reveals a significant performance gain for the relational models applied to the graphs generated by PSRB when compared to GCN, GAT and GraphSAGE models. Considering the dataset characteristics (Table 1), we observe that it contains a large number of disconnected components, which explains the effectiveness of the proposed rewiring-based model.

In the DBLP dataset (Table 4), even when using a shallow architecture, PSRB achieved the best results for the 0.01 and 0.05 rates, while deeper models outperformed it from the 0.1 rate onwards. Parametric testing of the RCSVM algorithm did not yield any valid results.

For the CiteSeer, Cora and PubMed datasets (Tables 5, 6, 7, respectively), the approaches based on rewiring and relational graphs also stood out when compared to existing methods in the literature. It is important to note that the proposed PSRB approach enables the use of relational models on non-relational data for PUL problems. In other words, the PSRB algorithm generates relations in the graphs, which facilitates message passing between vertices regardless of the GNN model employed, and enhances their performance as indicated by the experimental results.

Moreover, even with shallower neural networks (such as GCN), in most of the scenarios the PSRB model outperformed all state-of-the-art algorithms, including deeper networks that do not use sequential rewiring strategies. These experiments illustrate the significant gains provided by the proposed approach.

Table 3 Average *F1* score for Amazon photo dataset

Model/rate	0.01	0.05	0.10	0.15	0.20	0.25
CCRNE	0.4628 ± 0.01	0.5031 ± 0.03	0.4688 ± 0.02	0.4613 ± 0.04	0.4688 ± 0.05	0.4454 ± 0.02
LP_PUL	0.4538 ± 0.00	0.5106 ± 0.00	0.5054 ± 0.00	0.5162 ± 0.01	0.5263 ± 0.00	0.5195 ± 0.00
MCLS	0.4253 ± 0.00	0.4137 ± 0.00	0.401 ± 0.00	0.3896 ± 0.00	0.3767 ± 0.00	0.3651 ± 0.00
PU_LP	0.3655 ± 0.03	0.3732 ± 0.02	0.3805 ± 0.05	0.5092 ± 0.19	0.7258 ± 0.11	0.7888 ± 0.03
RCSVM	0.355 ± 0.02	0.3137 ± 0.01	0.2711 ± 0.03	0.2827 ± 0.02	0.2744 ± 0.02	0.2737 ± 0.01
GCN	0.4528 ± 0.03	0.4723 ± 0.04	0.4881 ± 0.06	0.4984 ± 0.02	0.4968 ± 0.01	0.5010 ± 0.02
GAT	0.4918 ± 0.06	0.5272 ± 0.04	0.5790 ± 0.05	0.6065 ± 0.02	0.5932 ± 0.08	0.5635 ± 0.05
GraphSage	0.2522 ± 0.30	0.6389 ± 0.06	0.5933 ± 0.07	0.4925 ± 0.28	0.5782 ± 0.06	0.5699 ± 0.04
PSRB (RGCN)	0.4900 ± 0.26	0.6374 ± 0.03	0.6484 ± 0.02	0.7085 ± 0.07	0.7317 ± 0.05	0.7273 ± 0.09
PSRB (RGAT)	0.5561 ± 0.11	0.6525 ± 0.03	0.6687 ± 0.01	0.7355 ± 0.07	0.6613 ± 0.03	0.7423 ± 0.06
PSRB (RGraphSage)	0.2101 ± 0.29	0.1385 ± 0.31	0.3651 ± 0.34	0.2695 ± 0.31	0.4712 ± 0.43	0.7932 ± 0.08

The best results are shown in bold

Table 4 Average *F1* score for DBLP dataset

Model/rate	0.01	0.05	0.10	0.15	0.20	0.25
CCRNE	0.5591 ± 0.02	0.5854 ± 0.01	0.5841 ± 0.00	0.6042 ± 0.02	0.6076 ± 0.04	0.6609 ± 0.04
LP_PUL	0.7111 ± 0.00	0.763 ± 0.02	0.7702 ± 0.02	0.7786 ± 0.01	0.7786 ± 0.00	0.7875 ± 0.01
MCLS	0.6059 ± 0.01	0.5787 ± 0.03	0.5623 ± 0.03	0.543 ± 0.03	0.5179 ± 0.02	0.5088 ± 0.02
PU_LP	0.1547 ± 0.03	0.2673 ± 0.03	0.3278 ± 0.04	0.423 ± 0.02	0.6056 ± 0.03	0.6719 ± 0.02
GCN	0.3807 ± 0.23	0.5391 ± 0.41	0.7906 ± 0.03	0.8152 ± 0.04	0.8155 ± 0.03	0.8079 ± 0.01
GAT	0.5934 ± 0.23	0.8368 ± 0.03	0.8441 ± 0.01	0.8394 ± 0.02	0.8352 ± 0.01	0.8265 ± 0.03
GraphSage	0.6709 ± 0.37	0.8295 ± 0.01	0.8304 ± 0.01	0.8219 ± 0.01	0.8163 ± 0.01	0.816 ± 0.0
PSRB (RGCN)	0.8088 ± 0.04	0.8488 ± 0.02	0.8487 ± 0.01	0.8477 ± 0.01	0.8469 ± 0.02	0.8475 ± 0.01
PSRB (RGAT)	0.7968 ± 0.04	0.8417 ± 0.02	0.8533 ± 0.01	0.8647 ± 0.0	0.8521 ± 0.01	0.8412 ± 0.02
PSRB (RGraphSage)	0.7982 ± 0.05	0.8389 ± 0.0	0.833 ± 0.03	0.8469 ± 0.01	0.8444 ± 0.02	0.8507 ± 0.01

The best results are shown in bold

Table 5 Average *F1* score for CiteSeer dataset

Model/rate	0.01	0.05	0.10	0.15	0.20	0.25
CCRNE	0.3454 ± 0.03	0.4655 ± 0.04	0.5797 ± 0.03	0.5911 ± 0.02	0.593 ± 0.02	0.5718 ± 0.06
LP_PUL	0.3293 ± 0.00	0.4076 ± 0.01	0.4214 ± 0.00	0.418 ± 0.00	0.4157 ± 0.00	0.4157 ± 0.00
MCLS	0.3075 ± 0.09	0.3666 ± 0.04	0.3694 ± 0.04	0.4323 ± 0.06	0.456 ± 0.03	0.4021 ± 0.0
PU_LP	0.0910 ± 0.03	0.1496 ± 0.02	0.2060 ± 0.01	0.2016 ± 0.01	0.2137 ± 0.03	0.2338 ± 0.02
GCN	0.3762 ± 0.03	0.4246 ± 0.02	0.4377 ± 0.02	0.4204 ± 0.01	0.3913 ± 0.04	0.4014 ± 0.01
GAT	0.3653 ± 0.02	0.4511 ± 0.05	0.4423 ± 0.05	0.4473 ± 0.03	0.4836 ± 0.05	0.4086 ± 0.04
GraphSage	0.3649 ± 0.09	0.4962 ± 0.06	0.6135 ± 0.05	0.5619 ± 0.05	0.5595 ± 0.05	0.5429 ± 0.04
PSRB (RGCN)	0.3823 ± 0.11	0.6248 ± 0.04	0.6066 ± 0.03	0.6472 ± 0.02	0.6643 ± 0.01	0.6628 ± 0.02
PSRB (RGAT)	0.4901 ± 0.12	0.626 ± 0.06	0.5958 ± 0.08	0.645 ± 0.03	0.6511 ± 0.03	0.652 ± 0.02
PSRB (RGraphSage)	0.4683 ± 0.12	0.6111 ± 0.05	0.6453 ± 0.03	0.6009 ± 0.06	0.6252 ± 0.05	0.6259 ± 0.02

The best results are shown in bold

Table 6 Average *F1* score for Cora dataset

Model/rate	0.01	0.05	0.10	0.15	0.20	0.25
CCRNE	0.3962±0.06	0.5627±0.02	0.5761±0.03	0.5845±0.02	0.6163±0.03	0.6198±0.03
LP_PUL	0.4688±0.01	0.5036±0.01	0.5422±0.02	0.5829±0.03	0.5997±0.02	0.6103±0.02
MCLS	0.4536±0.01	0.446±0.01	0.4404±0.02	0.4082±0.01	0.4061±0.03	0.3900±0.03
PU_LP	0.5739±0.10	0.6801±0.02	0.6716±0.02	0.6795±0.02	0.6677±0.01	0.6390±0.01
RCSVM	0.3723±0.12	0.4034±0.08	0.3689±0.04	0.3167±0.03	0.3344±0.03	0.3126±0.01
GCN	0.5839±0.09	0.6339±0.19	0.6786±0.03	0.6755±0.08	0.6666±0.04	0.6681±0.04
GAT	0.5710±0.09	0.7064±0.06	0.6833±0.07	0.6635±0.09	0.6984±0.05	0.6681±0.02
GraphSage	0.5895±0.12	0.7123 ±0.06	0.6779±0.05	0.6481±0.04	0.7336±0.02	0.6840±0.04
PSRB (RGCN)	0.5220±0.20	0.7102±0.03	0.7078 ± 0.05	0.6915±0.03	0.6909±0.03	0.7408 ± 0.01
PSRB (RGAT)	0.6941 ±0.01	0.6877±0.04	0.6937±0.04	0.6664±0.08	0.7149±0.02	0.7019±0.05
PSRB (RGraphSage)	0.5733±0.12	0.6898±0.02	0.7018±0.07	0.7237 ±0.03	0.7401 ±0.00	0.7350±0.05

The best results are shown in bold

Table 7 Average *F1* score for PubMed dataset

Model/Rate	0.01	0.05	0.10	0.15	0.20	0.25
CCRNE	0.6255 ± 0.03	0.7073 ± 0.01	0.7167 ± 0.01	0.7108 ± 0.03	0.6716 ± 0.07	0.6952 ± 0.02
LP_PUL	0.5778 ± 0.0	0.5821 ± 0.01	0.5867 ± 0.0	0.5849 ± 0.0	0.5845 ± 0.0	0.5808 ± 0.0
MCLS	0.5542 ± 0.01	0.5778 ± 0.06	0.5508 ± 0.06	0.5723 ± 0.03	0.5527 ± 0.05	0.5044 ± 0.03
PU_LP	0.3693 ± 0.02	0.4059 ± 0.03	0.5189 ± 0.07	0.6866 ± 0.04	0.7184 ± 0.0	0.7132 ± 0.03
RCSVM	0.4245 ± 0.01	0.3794 ± 0.01	0.3617 ± 0.01	0.3346 ± 0.01	0.3129 ± 0.0	0.2917 ± 0.01
GCN	0.6020 ± 0.01	0.6074 ± 0.02	0.5994 ± 0.01	0.5987 ± 0.02	0.5762 ± 0.01	0.5755 ± 0.02
GAT	0.6469 ± 0.01	0.6409 ± 0.02	0.6142 ± 0.01	0.6057 ± 0.02	0.6140 ± 0.02	0.6423 ± 0.05
GraphSage	0.6105 ± 0.03	0.5922 ± 0.01	0.5783 ± 0.00	0.5913 ± 0.04	0.5801 ± 0.04	0.5927 ± 0.03
PSRB (RGCN)	0.7089 ± 0.03	0.7898 ± 0.04	0.7679 ± 0.02	0.7205 ± 0.01	0.6944 ± 0.03	0.5643 ± 0.07
PSRB (RGAT)	0.7327 ± 0.03	0.8142 ± 0.01	0.7845 ± 0.03	0.7197 ± 0.04	0.6641 ± 0.04	0.6042 ± 0.04
PSRB (RGraphSage)	0.6788 ± 0.02	0.7922 ± 0.02	0.7686 ± 0.04	0.7139 ± 0.04	0.7142 ± 0.06	0.7046 ± 0.04

The best results are shown in bold

5 Conclusion

This article introduced the PSRB model, based on the two-step technique for PUL. The PSRB model applies a rewiring strategy to graphs, focusing exclusively on connections between positive vertices, increasing the graph's density and facilitating the propagation of information.

To handle the graphs created by the rewiring strategy, we developed a GNN to process information across multiple graphs. Negative data inference was performed by calculating the Euclidean distance between the examples most distant from the positive ones in the latent representation space. Finally, the inference of negative data was performed by a MLP coupled to the GNN model.

This work presents several contributions: It introduces the PSRB rewiring model for PUL tasks, proposes a relational approach to process sequentially generated graphs and incorporates a density function capable of effectively creating new connections among positive vertices. We conduct a parametric analysis to identify key relationships between the proposed parameters and the characteristics of the training data. To the best of our knowledge, this is the first time that rewiring models have been integrated with PUL problems using two-step techniques.

The experimental results showed that the proposed approach is competitive with state-of-the-art PUL methods and efficient at extracting negative data. Although parameterizing the probability equation presents a challenge, it is possible to verify relationships between the parameters and the statistical measures of the datasets, which can facilitate parameterization.

While this research provides a robust solution for PUL tasks, the algorithm is not limited to this paradigm. Future work will implement the rewiring strategy in multiclass problems and investigate additional model features, such as generating more representative embeddings. We also intend to investigate the effects of applying the rewiring strategy to negative samples, which may represent a more challenging problem.

Acknowledgements This research was funded by the Coordination for the Improvement of Higher Education Personnel (CAPES)—Code 001, through the provision of a master's scholarship, and by the São Paulo Research Foundation (FAPESP), Grant Number 22/03090-0, and by CNPq—National Council for Scientific and Technological Development, Grant Number 420025/2023-5.

Author Contributions The authors contributed equally to this work. G.M., A.V., and S.I. contributed to the design or implementation of the research, the analysis of the results, and the writing of the manuscript.

Data availability Data that support the findings of this study are available at <https://github.com/GHMessias/psrb>. The link is also available at the paper.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Bekker J, Davis J (2020) Learning from positive and unlabeled data: a survey. *Mach Learn* 109(4):719–760
2. Liu B, Dai Y, Li X, Lee WS, Yu PS (2003) Building text classifiers using positive and unlabeled examples. In: 3rd IEEE international conference on data mining, pp 179–186. IEEE
3. Claesen M, De Smet F, Suykens JA, De Moor B (2015) A robust ensemble approach to learn from positive and unlabeled data using SVM base models. *Neurocomputing* 160:73–84

4. Zhang Y, Ju X, Tian Y (2014) Nonparallel hyperplane support vector machine for PU learning. In: 2014 10th international conference on natural computation (ICNC), pp 703–708. IEEE
5. Shao Y-H, Chen W-J, Liu L-M, Deng N-Y (2015) Laplacian unit-hyperplane learning from positive and unlabeled examples. *Inf Sci* 314:152–168
6. Northcutt CG, Wu T, Chuang IL (2017) Learning with confident examples: rank pruning for robust classification with noisy labels. arXiv preprint [arXiv:1705.01936](https://arxiv.org/abs/1705.01936)
7. Elkan C, Noto K (2008) Learning classifiers from only positive and unlabeled data. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 213–220
8. Li X, Liu B (2003) Learning to classify texts using positive and unlabeled data. In: *IJCAI*, vol 3, pp 587–592. Citeseer
9. Liu L, Peng T (2014) Clustering-based method for positive and unlabeled text categorization enhanced by improved TFIDF. *J Inf Sci Eng* 30(5):1463–1481
10. Chaudhari S, Shevade S (2012) Learning from positive and unlabelled examples using maximum margin clustering. In: Proceedings of the 19th international conference on neural information processing (ICONIP 2012), Doha, Qatar, November 12–15, 2012, part III 19, pp 465–473. Springer, Berlin
11. Ma S, Zhang R (2017) Pu-Lp: a novel approach for positive and unlabeled learning by label propagation. In: 2017 IEEE international conference on multimedia & expo workshops (ICMEW), pp 537–542. IEEE
12. Carnevali JC, Rossi RG, Milios E, Andrade Lopes A (2021) A graph-based approach for positive and unlabeled learning. *Inf Sci* 580:655–672
13. Li W, Zhang X-Y, Bao H, Shi H, Wang Q (2022) Prograph: robust network traffic identification with graph propagation. *IEEE/ACM Trans Netw* 31(3):1385–1399
14. Deng L, Liu X-Y, Zheng H, Feng X, Chen Z (2023) Graph-tensor neural networks for network traffic data imputation. *IEEE/ACM Trans Netw* 31(6):3010–3024
15. Sahay R, Nicoll S, Zhang M, Yang T-Y, Joe-Wong C, Douglas KA, Brinton CG (2023) Predicting learning interactions in social learning networks: a deep learning enabled approach. *IEEE/ACM Trans Netw* 31(5):2086–2100
16. Ferriol-Galmés M, Paillisse J, Suárez-Varela J, Rusek K, Xiao S, Shi X, Cheng X, Barlet-Ros P, Cabellos-Aparicio A (2023) Routenet-Fermi: network modeling with graph neural networks. *IEEE/ACM Trans Netw* 31(6):3080–3095. <https://doi.org/10.1109/TNET.2023.3269983>
17. Yu Z, Jin D, Liu Z, He D, Wang X, Tong H, Han J (2022) Embedding text-rich graph neural networks with sequence and topical semantic structures. *Knowl Inf Syst* 65(2):613–640. <https://doi.org/10.1007/s10115-022-01768-4>
18. Wu M, Pan S, Du L, Zhu X (2021) Learning graph neural networks with positive and unlabeled nodes. *ACM Trans Knowl Discov Data (TKDD)* 15(6):1–25
19. Schlichtkrull M, Kipf TN, Bloem P, Van Den Berg R, Titov I, Welling M (2018) Modeling relational data with graph convolutional networks. In: Proceedings of the 15th international conference on the semantic web (ESWC) 2018, Heraklion, Crete, Greece, June 3–7, 2018, vol 15, pp 593–607. Springer
20. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907)
21. Liu Y, Jin M, Pan S, Zhou C, Zheng Y, Xia F, Yu PS (2023) Graph self-supervised learning: a survey. *IEEE Trans Knowl Data Eng* 35(6):5879–5900. <https://doi.org/10.1109/TKDE.2022.3172903>
22. Ju W, Fang Z, Gu Y, Liu Z, Long Q, Qiao Z, Qin Y, Shen J, Sun F, Xiao Z, Yang J, Yuan J, Zhao Y, Wang Y, Luo X, Zhang M (2024) A comprehensive survey on deep graph representation learning. *Neural Netw* 173:106207. <https://doi.org/10.1016/j.neunet.2024.106207>
23. Chen C, Wu Y, Dai Q, Zhou HY, Xu M, Yang S, Han X, Yu Y (2022) A survey on graph neural networks and graph transformers in computer vision: a task-oriented perspective. arXiv preprint [arXiv:2209.13232](https://arxiv.org/abs/2209.13232)
24. Gao C, Wang X, He X, Li Y (2022) Graph neural networks for recommender system. In: Proceedings of the 15th ACM international conference on web search and data mining, pp 1623–1625
25. Li H, Zhao Y, Mao Z, Qin Y, Xiao Z, Feng J, Gu Y, Ju W, Luo X, Zhang M (2024) A survey on graph neural networks in intelligent transportation systems. arXiv preprint [arXiv:2401.00713](https://arxiv.org/abs/2401.00713)
26. Lin X, Quan Z, Wang ZJ, Ma T, Zeng X (2020) KGNN: knowledge graph neural network for drug–drug interaction prediction. *IJCAI* 380:2739–2745
27. Bruna J, Zaremba W, Szlam A, LeCun Y (2013) Spectral networks and locally connected networks on graphs. arXiv preprint [arXiv:1312.6203](https://arxiv.org/abs/1312.6203)
28. Yuan J, Luo X, Qin Y, Zhao Y, Ju W, Zhang M (2023) Learning on graphs under label noise. In: 2023 IEEE international conference on acoustics, speech and signal processing (ICASSP 2023), pp 1–5. <https://doi.org/10.1109/ICASSP49357.2023.10096088>
29. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. arXiv preprint [arXiv:1710.10903](https://arxiv.org/abs/1710.10903)

30. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: *Advances in neural information processing systems*, vol. 30 (2017)
31. Akansha, S.: Over-squashing in graph neural networks: a comprehensive survey. *Neurocomputing* **2025**:130389 (2025)
32. Gabriëlsso, R.B., Yurochkin, M., Solomon, J.: Rewiring with positional encodings for graph neural networks. *Transactions on Machine Learning Research* (2023)
33. Abboud, R., Dimitrov, R., Ceylan, I.I.: Shortest path networks for graph property prediction. In: *Learning on graphs conference*, PMLR, pp. 5–1 (2022).
34. Cai, C., Hy, T.S., Yu, R., Wang, Y.: On the connection between mpnn and graph transformer. In: *International Conference on Machine Learning*, PMLR, pp. 3408–3430 (2023).
35. Kreuzer, D., Beaini, D., Hamilton, W., L'etourneau, V., Tossou, P.: Rethinking graph transformers with spectral attention. In: *Advances in Neural Information Processing Systems*, vol. 34, pp. 21618–21629 (2021)
36. Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., Liu, T.-Y.: Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems* vol. 34, pp. 28877–28888 (2021)
37. Barbero, F., Vellingker, A., Saberi, A., Bronstein, M.M., Giovanni, F.D.: Locality-aware graph-rewiring in gnns. *CoRR* abs/2310.01668 (2023)
38. Arnaiz-Rodríguez, A., Begga, A., Escolano, F., Oliver, N.M.: Diffwire: Inductive graph rewiring via the lov'asz bound. In: *The First Learning on Graphs Conference* (2022)
39. Black, M., Wan, Z., Nayyeri, A., Wang, Y.: Understanding oversquashing in gnns through the lens of effective resistance. In: *International Conference on Machine Learning*, PMLR, pp. 2528–2547 (2023)
40. Berahmand K, Daneshfar F, Salehi ES, Li Y, Xu Y (2024) Autoencoders and their applications in machine learning: a survey. *Artif Intell Rev* 57(2):28
41. Kipf, T.N., & Welling, M.: Variational graph auto-encoders. In: *Bayesian Deep Learning Workshop NIPS* (2016), Barcelona, Spain. <https://arxiv.org/abs/1611.07308v1>
42. Zhou, D., Bousquet, O., Lal, T., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: *Adv Neural Inf Process Syst*, vol. 16, pp. 321–328. MIT Press, Vancouver, British Columbia, Canada (2003)
43. Rusch TK, Bronstein MM, Mishra S (2023) A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*
44. Karhadkar, K., Banerjee, P., Montufar, G.: Fossr: First-order spectral rewiring for addressing oversquashing in gnns. In: *International Conference on Learning Representations* (2023)
45. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. *J Mach Learn Res* 12:2825–2830
46. Yang, Z., Cohen, W., Salakhudinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Balcan, M.F., Weinberger, K.Q. (eds.) *Proceedings of The 33rd International Conference on Machine Learning*. *Proceedings of machine learning research*, vol. 48, pp. 40–48. PMLR, New York, New York, USA (2016)
47. Bojchevski, A., Günnemann, S.: Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In: *International Conference on Learning Representations* (2018)
48. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018)
49. Fey M, Lenssen JE (2019) Fast graph representation learning with PyTorch geometric. In: *ICLR workshop on representation learning on graphs and manifolds*
50. Rong Y, Huang W, Xu T, Huang J (2020) Dropedge: towards deep graph convolutional networks on node classification. In: *International conference on learning representations*. <https://openreview.net/forum?id=Hkx1qkrKPr>
51. Lu W, Zhan Y, Lin B, Guan Z, Liu L, Yu B, Zhao W, Yang Y, Tao D (2024) Skipnode: on alleviating performance degradation for deep graph convolutional networks. *IEEE Trans Knowl Data Eng* 36(11):7030–7043. <https://doi.org/10.1109/TKDE.2024.3374701>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.