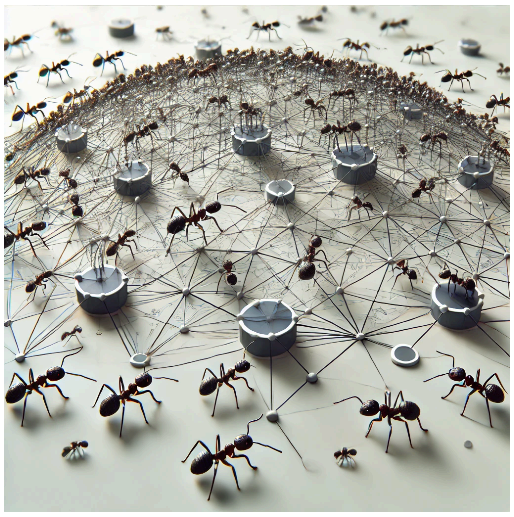
Capstone Project Phase A 24-2-D-19

Citation Prediction using Ant Colony based Multi-Level Network Embedding



Yoni Azeraf - 209459239

Itamar Kraus - 318304763

Supervisor:

Dvora Toledano Kitai

GitHub

Table of Content

Abstract. This project addresses the issue of including citations that are not necessarily relevant or have a weak and not necessarily significant connection to the paper. This issue has a negative effect on the quality and relevance of papers and on the ability of the readers in navigating the sea of academic information. Existing approaches to tackle this problem, such as manual screening and expert reviews do exist, but these methods suffer from subjectivity and inadequate accuracy. By utilizing Ant Colony based Multi-level Network Embedding (ACE) and gaining insights from "ACE: Ant Colony based Multi-level Network Embedding for Hierarchical Graph Representation Learning" [11]. This project introduces a different approach. It involves creating a citation graph and utilizing an Ant Colony Optimization (ACO) method [7] based algorithm to identify those citations, enabling the assessment of their relevance. This method aims to improve the accuracy and objectivity in evaluating the relevance of citations in academic papers.

Keywords: Network Embedding · Ant Colony Optimization · Coarsening · Academic Citation · Node Classification.

1. Introduction

Graph analysis is a powerful way to represent detailed data from sources like social networks, biological networks, and the World Wide Web.

By analyzing graphs, we can gain valuable insights on the underlying structures, patterns, and hidden reveal relationships within complex networks.
Clustering is a common task in graph analysis which helps to simplify the complexity of networks by highlighting the natural groupings.

ACE [11] is an innovative algorithm that stands out by using ant colony principles to perform multi-level network embedding, capturing the clustering properties of graphs at multiple levels.

Unlike traditional network embedding algorithms that focus solely on the immediate neighbors of nodes, ACE employs a unique ant colony-based coarsening algorithm. This approach converts the graph into a series of multilevel clustering pyramids, allowing it to grasp the global clustering structures at various scales.
Additionally, ACE uses an ant colony-based random walk algorithm to explore the strength of relationships between nodes, especially within loop structures in the graphs.

2. Literature Review

2.1 Naive Way

A basic approach to represent nodes in a graph is by using the row vectors from the adjacency matrix as their feature vectors (as used in [2], [13]).
An adjacency matrix is a square matrix used to represent a graph, where each row vector corresponds to a node and the elements indicate the presence of edges between nodes.
That approach often becomes impractical for real-world graphs, which can have billions of nodes and edges. The resulting feature vectors are typically high-dimensional and sparse, leading to poor performance in classifiers.

2.2 Embedding Approach

This approach's goal is to represent nodes using lower-dimensional embedding vectors, allowing the graph to be mapped to an embedding vector space.
Nodes with close relationships will be mapped to similar embedding vectors.
This space can be processed using Machine Learning (ML) methods for tasks such as link prediction, node classification, visualization, etc.

2.2.1 Embedding algorithms based on matrix factorization

Embedding algorithms based on matrix factorization are effective for processing small graphs to produce low-dimensional node vectors, but they struggle to scale up for handling large-scale graphs.

2.2.1.1 Locally Linear Embedding

Locally Linear Embedding (LLE) method [16] preserves local relationships by reconstructing each node based on its nearest neighbors, effectively capturing the local structure within the graph.

2.2.1.2 Laplacian Eigenmaps

By focusing on the graph Laplacian, Laplacian Eigenmaps approach [1] seeks to maintain the local connectivity of nodes, making it excellent for highlighting the inherent geometry of the data.

2.2.1.3 GraRep

GraRep technique [2] extends beyond immediate neighborhoods by considering different powers of the adjacency matrix to capture various scales of node relationships within its embeddings.

2.2.1.4 HOPE

HOPE procedure [13] efficiency preserves higher-order proximities by leveraging a similarity matrix, which allows it to capture the asymmetry in node relationships effectively.

2.2.2 Random walk-based network embedding algorithms

Random walk-based network embedding algorithms are more adept at addressing the scalability issues associated with large graphs.

2.2.2.1 DeepWalk

DeepWalk technique [14] uses random walks on a graph to generate node sequences, applying natural language processing techniques to create vector representations that capture the network's structure.

2.2.2.2 Node2Vec

Node2Vec method [8] flexibly learns node representations by efficiently balancing between exploring local neighborhoods and sampling more distant parts of the graph.

2.2.2.3 LINE

LINE approach [18] designed to handle very large networks by preserving both first order (direct neighbors) and second order (neighbors of neighbors) proximity.

2.2.3 Deep learning-based network embedding algorithms

Deep learning-based network embedding algorithms utilize complex models with larger capacities to uncover deeper properties of nodes.

2.2.3.1 SDNE

SDNE mechanism [19] uses a semi-supervised deep model that preserves both the first order and second-order proximities between nodes, enhancing its ability to capture the complex structures within the graph.

2.2.3.2 GCN

GCN [6], [10] method applies convolutional neural network concepts to graph data, aggregating features from a node's neighbors to learn powerful representations that incorporate both local graph structure and node features.

2.2.3.3 GAN

Building on the mechanism of attention, GAN approach [5],[20] allows nodes to weigh the importance of their neighbors' features dynamically, leading to more precise and context-aware embeddings.

Nevertheless, the above Embedding algorithms represent the local structures surrounding a node and fail to reflect the global hierarchical clustering properties within a graph.

2.3 Advanced Embedding Algorithms

These algorithms aim to deal with the problem of representing the global structure of a network.

2.3.1 Walklets

Walklets technique [15] selectively skipping nodes in random walk sequences to cover various graph scales, enhances the ability to grasp different levels of node relationships by varying the length and connectivity of paths in its random walks.

2.3.2 HARP

HARP [3] is a hierarchical embedding method designed to encapsulate the characteristics of large-scale structures. It progressively simplifies a graph by randomly merging connected nodes into a sequence of coarser graphs, from which it then recursively constructs embedding vectors. However, the random merging process may not accurately capture the intrinsic clustering structures of the graph.

2.3.3 NetLay

A Graph Network layering algorithm [9] that simplifies complex graph networks by structuring them into hierarchical layers. Identifies using random walks and groups interconnected communities within the graph, then merges these into super-nodes and establishes new interconnections that preserve the network's essential structure.

The approach of uncovering the hierarchical structure by constructing a layered pyramid of the graph also characterizes ACE [11] algorithm employed in this project.

3. Background

3.1 Preliminaries

3.1.1 Problem definition of network embedding

Given a Graph $G = < V, E >$, the objective of network embedding is to transform each node $V_i \in V$ into a dimensional vector $v_i \in R^d (d \ll |V|)$ known as an embedding vector.

Similarity between any two embedding vectors v_i and v_j reflects the relationship between the corresponding nodes V_i and V_j.

This approach allows the nodes of a graph to be represented in a feature matrix $M_{d\times|V|}$, which can then be utilized for tasks such as classification, clustering, and prediction.

3.1.2 Graph Clustering Pyramid

To capture the hierarchical clustering property, a graph is modeled by a Graph Clustering Pyramid.

Graph Clustering Pyramid $\Psi_G = \{G_0, G_1, ..., G_L\}$ where G_0 is the original graph and,

G_{l+1} $(0 \le l < L)$ is the clustering result of G_l.

In this model, clustering the nodes simplifies the graph into multiple layers. Nodes in higher layers correspond to clusters in the layers below, revealing complex structures through multi-layer decomposition. Higher-level graphs clarify the relationships between clusters and display global structures, whereas lower layers provide detailed local information.

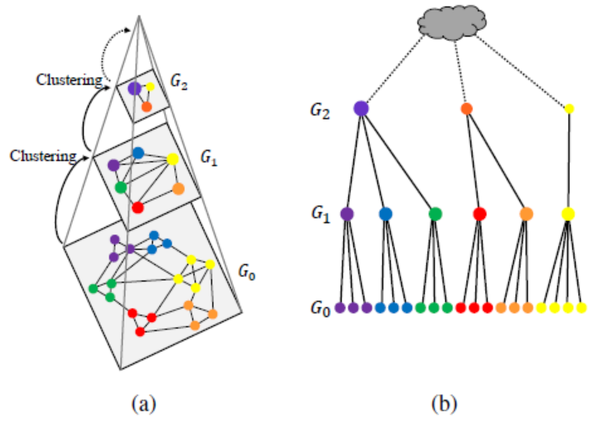
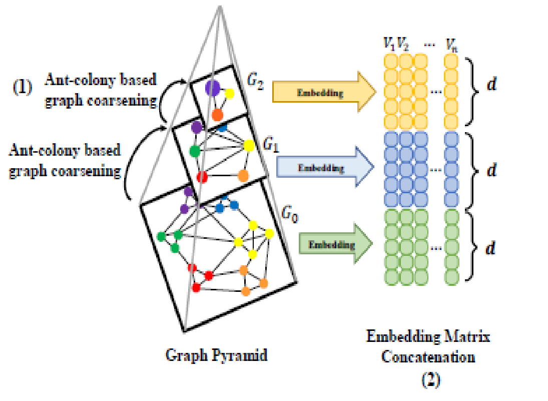


Figure 1: (a) Graph Clustering Pyramid
(b) Clustering tree of nodes

3.2 Multi Level Network Embedding

Algorithm 1 - MLNE:

Input: Graph $G(V, E)$, embedding algorithm EMB,
Embedding representation size d
Output: Node representation vector embedding matrix $\Phi \in R^{|V|\times d\cdot L}$



(1) Ant-colony based graph coarsening
Ant-colony based graph coarsening
G_2
G_1
G_0
Embedding
Embedding
Embedding
$V_1 V_2 \cdots V_n$
d
d
d
Graph Pyramid
Embedding Matrix Concatenation (2)

Figure 2: Multi-Level Network Embedding Framework

```
#Apply Ant colony graph coarsening on the graph to receive the graph layers
1. $\{G_0, G_1, ..., G_L\} \leftarrow ACGraphCoarsening(G)$
2. For each layer $G_i \in \{G_0, G_1, ..., G_L\}$ do
    #Apply the embedding method on the layer
    2.1. $\Phi_i \leftarrow EMB(G_i, d)$
3. $\Phi \leftarrow \{\Phi_0, \Phi_1, ..., \Phi_L\}$
4. Return $\Phi$
```

3.3 Ant Colony Based Graph Coarsening

The fundamental process in building a Graph Clustering Pyramid involves clustering closely related nodes and merging them into coarser graphs in the upper layers.
The main challenge lies in determining the relationship between nodes in a graph

In an undirected graph, the local context of two nodes can be described by a set of closed loops.
A higher number of loops that include both nodes suggests a stronger relationship.
Similarly, smaller loops indicate a closer relationship between the nodes.

To identify the close connections defined by loops, it is suggested to use an ant colony optimization (ACO) [7] based *Ant Colony Walking* algorithm.

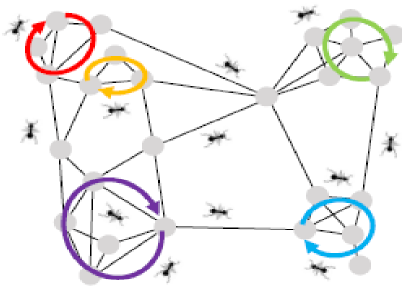


Figure 3: Ant walking in the graph

Initially, several ants are released from every node in the graph
to start their random walks.
The number of ants released is directly related to the degree of the initial node.
These ants have the goal of identifying loops within the graph and when a loop is found, each ant scatters pheromones throughout the loop to mark a successful find.
The amount of pheromone added to each edge is inversely proportional with the loop's length.
Therefore, edges that are part of many short loops end up with higher pheromone levels, which highlights the strength of the connection between two nodes.

3.3.1 Ant Colony Walking

As mentioned in the previous section we utilize a random walking approach to identify the close connections defined by loops.

The *Ant Colony Walking* algorithm (Algorithm 2) receives a graph as the input.
First, There is an initialization of the transition probabilities P between every two nodes u_i, u_j as the weight of the edge between them divided by the sum of the weights of all edges originating from node u_i, meaning $P(u_i \rightarrow u_j) = \frac{W_{ij}}{\Sigma_k W_{ik}}$.
The algorithm is occurring k times (hyper parameter).
For every node in the graph, it's degree is being calculated and sets $n * degree$ ants to start walking from this node.
Each ant's purpose is to discover loops in the graph.
The maximum length of a loop that can be discovered is defined by the hyper parameters max_steps.
Once a loop is detected (the ant went back to her starting node), the ant will travel again through that loop and scatter pheromone on its edges in a way that the amount of pheromone scattered is inversely proportional with the loop's length (the amounts will be saved in a matrix).
On each one of the k iterations the algorithm recalculates the transition probabilities by considering the amount of pheromone that the edges had (using a hyper-parameter α which is used to adjust the importance of edge pheromone)
$P(u_i \rightarrow u_j) = \frac{W_{ij} \cdot \rho^{\alpha}_{ij}}{\Sigma_k W_{ik} \cdot \rho^{\alpha}_{ik}}$, by doing so ants will focus on the edges that are part of more loops signifying a closer relationship.
At the end of all iterations the pheromone matrix is returned.

<u>Algorithm 2 - Ant Colony Walking (ACWalk):</u>
<u>Input</u>: Graph $G(V, E)$
<u>Output</u>: Graph edge pheromone matrix ρ

```
1. Initialize the transition probabilities

2. Repeat $k$ (hyper parameter) iterations

    2.1. For every node $V_{init} \in V$
        2.1.1 $d_{init} \leftarrow Degree(V_{init})$  # $d_{init}$ represent its degree.
        2.1.2. Repeat $|V| * d_{init}$ iterations  #Simulate $|V| * d_{init}$ ant walks
            2.1.2.1. $path \leftarrow \{V_{init}\}$ .
            2.1.2.2. While $|path| < max\_steps$ (hyper parameter) do

                2.1.2.2.1. Set $V_{next}$ to the next node according the
                           transition probabilities

                2.1.2.2.2. $path \leftarrow path \cup \{V_{next}\}$.
                2.1.1.2.3.  If $v_{next} == V_{init}$ then
            2.1.1.2.3.1. For consecutive nodes $V_i$, $V_j$ in the
                         $path$ array set
                         $\rho_{ij} \leftarrow \rho_{ij} + \frac{1}{length(path)}$
    2.2 Update the transition probabilities
3. Return ρ
```

3.3.2 Adaptive Threshold Selection

Based on the observations of ant-colony walking, It was noticed in [11] that the pheromone on the edges of real-world graphs typically forms a distinct segmented distribution.
Some edges gather significantly more pheromone than others, allowing to classify them into two clear categories:
edges that indicate strong relationships and serve as internal links within the clusters, and edges that indicate weak relationships and act as links between different clusters.

To construct the Graph Clustering Pyramid, we cluster nodes with strong relationships into a single super node in the upper layer graphs.
A straightforward method is to merge nodes connected by edges with pheromone levels above a certain threshold.

This threshold is determined by an *Adaptive Threshold Selection* (Algorithm 3).

The algorithm gets as input the graph edge pheromone matrix ρ from algorithm 2. first, excluding the edges that does not contain any pheromone and saving it in a list ρ' and the length of that list in l.

After that, sorting the list ρ' in descending order and "draw" a line between two points.
one point is the "highest" point in the graph and the other one is the "lowest" in the graph. (The graph always has a shape that resembles the "L" letter).
Then iterate over the points and find the elbow point which is the point that has the maximum distance from the line, the Y value, amount of pheromone, will be selected as the threshold.

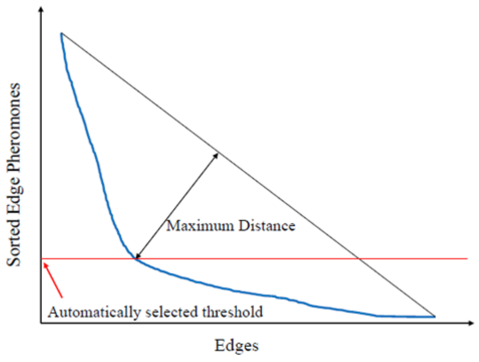


Figure 4: Select the threshold automatically according to the sorted edge pheromones distribution.

Algorithm 3 - Adaptive Threshold Selection (ATS):
Input: Graph edge pheromone matrix ρ
Output: $threshold$

```
    #Set a list ρ' as the list of all the positive pheromone edges in the matrix.
    1. $\rho' \leftarrow (\rho_{ij} \mid \rho_{ij} > 0\}$
2. $l \leftarrow length(\rho')$
3. Sort the list ρ' in descending order ($\rho'_1$ as the largest amount of pheromone, $\rho'_l$
   as the lowest amount)
4. Connect the points $(1,\ \rho'_1)$ and $(l,\ \rho'_l)$ by a $line$
5. For each point $(i, \rho'_i)$ do
   #Calculate the distance of the point to the line
   5.1. $distance_i \leftarrow CalcDistance((i, \rho'_i), line)$
#Find i of the maximum distance to the line
6. $i \leftarrow max(\{distance_0,\ distance_1,\ ..., distance_l\})$

7. Return $\rho'_i$
```

3.3.3 Graph clustering pyramid building

After the threshold is selected, we can introduce the algorithm *Graph clustering pyramid building* (Algorithm 4).

The algorithm shows the building process of the clustering pyramid.
The algorithm receives as an input a Graph G, sets L to be 0 and G_0 to be G.

There is a $ratio$ (hyper parameter) that determines when the algorithm stops to create more graph layers, it's a relation between the number of nodes and edges in the last coarsened graph (L) to the number of nodes and edges in the original graph.

each layer is created by the following steps:
first, executing Algorithm 2 to get the pheromone edges matrix and then executing Algorithm 3 to determine the threshold, after that the last layer is being coarsened into a new layer based on the threshold.
At the end the layers of the pyramid are returned.

Algorithm 4 - Graph clustering pyramid building:

Input: Graph $G(V, E)$

Output: Graph Clustering Pyramid Ψ_G

```
1. $L \leftarrow 0$
2. $G_0 \leftarrow G$
3. While $\frac{|V_L|}{|V|} > ratio$ AND $\frac{|E_L|}{|E|} > ratio$ do
   3.1. $\rho \leftarrow ACWalk(G_L)$ # Algorithm 2
   3.2. $threshold \leftarrow ATS(\rho)$ # Algorithm 3
   3.3. $G_{L+1} \leftarrow$ Coarsen $G_L$ based on the $threshold$
   3.4. $L \leftarrow L + 1$
```

```
4. $\Psi_G \leftarrow \{G_0, G_1, \ldots, G_L\}$
5. Return $\Psi_G$
```

3.4 Analysis of graph pyramid construction methods

ACO based graph coarsening algorithm (Algorithm 4) is not the only way to achieve the multi-level structures of graphs.

One way is using the random graph coarsening method which is adopted by HARP [3].
On each iteration the method randomly merges connected nodes to coarsen simplified graphs.
By doing it randomly, the nodes belonging to the same super node in the upper layer graph do not necessarily have strong relation to each other.
Thus, HARP is limited to capture the inner clustering structures within the graph.

Another popular way is community detection [4], [17].
This method's goal is to discover and merge groups of nodes (communities) that have more internal links than external links.
By doing so, it's still not guaranteed that each pair of nodes inside a community has equivalent strength.

In this project we suggested using the method for multi-level network embedding (MLNE), the nodes that are merged into the same super node are assigned with the same embedding vector in the upper layer.
With this approach the nodes with stronger relation will be merged, which can cause better results compared to the methods that were mentioned earlier.

3.5 Multi-Level Embedding Encoding

In this approach (MLNE) each graph G_x $(0 \leq x \leq L)$ in the Graph Clustering Pyramid
$\Psi_G = \{G_0, G_1, \ldots, G_L\}$
serves as a layer, with nodes in higher layers representing clusters of nodes from the lower layers.

Nodes in the base layer G_0 are denoted as $V_j^{(0)}$ and their corresponding ancestor nodes in a higher layer x are represented as $V_{Pj}^{(x)}$ where P represents the parent of j.

Respectively, the embedding vectors for these nodes are $v_j^{(0)}$(for the base layer G_0) and $v_{Pj}^{(x)}$ (for its corresponding ancestor in higher layer x).
The embedding vector $v_{Pj}^{(x)}$ is called the x-layer embedding vector of $V_j^{(0)}$.

These vectors are arranged into layer-specific matrices M^x, where each matrix dimension is $d\times|V_0|$, capturing the embeddings across all layers.
These matrices are then concatenated to form a single mixed embedding matrix $M_{dL\times|V_0|}$ representing the combined embeddings.

This structured approach allows integration with standard graph embedding algorithms, enabling the generation of node embeddings without modifying the original algorithms.

Thus, ACE [11] offers flexibility and ease of extension in applying different embedding techniques.

4. Expected Achievements

Our project's main goal is utilizing an Ant Colony based Multi-level Network Embedding (ACE [11]) to identify irrelevant citations in scholarly articles by capturing the clustering properties of graphs at multiple levels. By creating a citation graph from the Cora dataset [12] and implementing the ACE algorithm, we aim to identify and evaluate citation relevance.
Our expected achievement for this project is to expose at least 85% of the irrelevant citations within the citation graph.

5. Proposed Research Plan

In order to detect irrelevant citations we propose the following research plan.

Initially, we will utilize a Graph $G(V, E)$ based on the CORA dataset (collected by [12] from the Core website), in this graph the nodes V represent articles and the edges E represent the citations.

Our proposed method will run K iterations as follow:
First, We will randomly remove a predetermined percentage of edges from the graph $G(V, E)$, and create the modified graph $G'(V, E')$.
Then, we will execute Algorithm 1 - MLNE on the modified graph G', as previously

mentioned in section 3.2 we will construct the graph pyramid and then apply the chosen embedding method, node2vec [8], on each layer separately to create the concatenated embedding matrix $\Phi \in R^{|V| \times d \cdot L}$. In the concatenated embedding matrix, the columns represent $|V|$ multi-layered embedding vectors, i.e., the embedding of a node across all layers of the pyramid.
Afterwards, we will calculate the cosine similarity between every two embedding vectors in the matrix and normalize it to the range $[0, 1]$ using the sigmoid function σ. Two different nodes will be considered close if their normalized cosine similarity is below $threshold\ t_1$.
The results will be placed in a binary matrix $M_{|V| \times |V|}$, which will represent for each pair of nodes (V_i, V_j) in the graph whether they considered close (will be flagged as 0) or not (will be flagged as 1), the matrix obtained in iteration $k \mid 1 \le k \le K$ will be saved as M_k .

At the end of the K iterations, we will create statistical information for each pair of nodes (V_i, V_j) in the graph based on the times the pair considered close over the K iterations into a new matrix M_{stat}.

$$M_{stat} = \begin{bmatrix} 0 & s_{12} & \cdots & s_{1 \cdot |V|} \\ s_{12} & 0 & \cdots & s_{2 \cdot |V|} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1 \cdot |V|} & s_{2 \cdot |V|} & \cdots & 0 \end{bmatrix}$$

s_{ij} will represent the statistical similarity between nodes (V_i, V_j) and will be calculated by summing all their closeness results divided by the total number of iterations made (K), then convert the outcome into percentage, meaning

$$s_{ij} = \frac{\sum_{1}^{K} (M_k)_{i,j}}{K} \cdot 100 \ .$$

Any of the matrices $M_k \mid 1 \le k \le K$ and the statistical matrix M_{stat} are symmetric. The main diagonal of these matrices contains zeroes, which represent the closeness of a node to itself. Naturally, since a node is identical to itself, the value is zero.

The statistical matrix and a predefined threshold t_2 allows us to discern which nodes are closer to each other and which are less so.

A pair of nodes (V_i, V_j) with their corresponding statistical similarity value s_{ij} above the threshold t_2 indicates that they have strong relation and, vice versa, value s_{ij} below t_2 indicates that those pairs of nodes have a weak relation.

Based on these metrics, nodes in the graph which are linked by an edge e that have a weak relation, e will be considered irrelevant. Conversely, if the relation between the nodes is strong, e is likely to be significant.

Consequently, we can refine the CORA graph $G(V, E)$ and create a new graph $G_R(V, E_R)$ by removing irrelevant edges (citations), thereby ensuring the graph G_R reflects only strong, reliable connections between nodes.

Proposed research plan Pseudo-Code (ResearchPlanAlg) :

Input: CORA Graph $G(V, E)$, embedding algorithm EMB,
Embedding representation size d, Threshold t_1 ,Threshold t_2 , Percentage p,
Number of iterations K
Output: Refined Graph $G_R(V, E_R)$

```
1. For i from 1 to $K$ do:
    #Remove a certain percentage of the edges randomly and obtain the modified graph
    1.1 $G' \leftarrow RemoveEdges(G,\ p)$
    #Apply MLNE Algorithm and obtain the concatenated embedding matrix
    1.2 $\Phi' \leftarrow MLNE(G',\ EMB,\ d)$
    #Calculate the cosine similarity between every two embedding
    Vectors of the matrix and normalize it using sigmoid function, then check the
    closeness based on $t_1$ and save the corresponding flag in the new matrix
    1.3 For $m$ from 1 to $|V|$ do
        1.3.1 For $n$ from $m$ to $|V|$ do
            1.3.1.1 $res \leftarrow \sigma\left(\frac{\vec{v}_m \cdot \vec{v}_n}{|\vec{v}_m| \cdot |\vec{v}_n|}\right)$
            1.3.1.2 If $res < t_1$ then
                1.3.1.2.1 $(M_i)_{m,n} \leftarrow 0$
            1.3.1.3 If $res \geq t_1$ then
```

```
                    1.3.1.3.1 $(M_i)_{m,n} \leftarrow 1$

#Calculate the statistical matrix
2. For $m$ from 1 to $|V|$ do
    2.1 For $n$ from $m$ to $|V|$ do
        2.1.1 $(M_{stat})_{i,j} \leftarrow \frac{\sum_{1}^{K}(M_k)_{i,j}}{K} \cdot 100$

#Copy the CORA graph into a new graph
3. $G_R \leftarrow G(V, E)$

#Iterate over each value in the average matrix
4. For each $value_{m,n}$ in $M_{stat}$ do
    #If the value is above the threshold
    4.1 If $value_{m,n} < t_2$ then
        #Remove the edge between the corresponding nodes
        4.1.1 $G_R \leftarrow RemoveEdge(G_R, m, n)$

5. Return $G_R$
```

6. Evaluation/Verification Plan

To assess the proposed method, a predetermined percentage of edges will be added randomly to the CORA based graph. These added edges will be then saved as e'. The research plan will then be executed on the modified graph, and its output will be analyzed. Since the edges are added randomly, they are likely to connect vertices with weak relationships. As a result, we expect the suggested algorithm to remove these edges. The success of the proposed model can thus be evaluated by the percentage of edges from e' that are absent in the algorithm's final output. To accurately assess the success of the proposed model, we will execute the evaluation plan multiple times and then average the results.

Evaluation plan Pseudo-Code:

Input: CORA Graph $G(V, E)$, embedding algorithm EMB,
    Embedding representation size d, Threshold t_1, Threshold t_2, Percentage p,
    Number of iterations K
Output: $SuccessRate$

```
#Adding a certain percentage of the edges randomly and obtain the modified graph
1. $G', e' \leftarrow AddingEdges(G,\ p)$
2. $G_R \leftarrow ResearchPlanAlg(G',\ EMB,\ d,\ t_1,\ t_2,\ p,\ K)$
3. $Count \leftarrow 0$
4. For edge $e$ in $e'$ do
        3.1 If $e \notin E_R$ then
                3.1.1 $Count \leftarrow Count\ +\ 1$
5. $SuccessRate \leftarrow \frac{Count}{|e'|} \cdot 100$
6. Return $SuccessRate$
```

7. References

[1] Belkin, M., & Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14.

[2] Cao, S., Lu, W., & Xu, Q. (2015, October). GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international conference on information and knowledge management* (pp. 891-900).

[3] Chen, H., Perozzi, B., Hu, Y., & Skiena, S. (2018, April). Harp: Hierarchical representation learning for networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).

[4] Cordasco, G., & Gargano, L. (2010, December). Community detection via semi-synchronous label propagation algorithms. In *2010 IEEE International Workshop on Business Applications of Social Network Analysis (BASNA)* (pp. 1-8). IEEE.

[5] Dai, Q., Li, Q., Tang, J., & Wang, D. (2018, April). Adversarial network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).

[6] Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, 29.

[7] Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3), 243-278.

[8] Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 855-864).

[9] Jin, H., Xu, G., Cheng, K., Liu, J., & Wu, Z. (2022). A link prediction algorithm based on GAN. Electronics, 11(13), 2059.

[10] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[11] Lv, J., Zhong, J., Liang, J., & Yang, Z. (2019). ACE: Ant colony based multi-level network embedding for hierarchical graph representation learning. *IEEE Access*, 7, 73970-73982.

[12] McCallum, A. K., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, 3, 127-163.

[13] Ou, M., Cui, P., Pei, J., Zhang, Z., & Zhu, W. (2016, August). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1105-1114).

[14] Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 701-710).

[15] Perozzi, B., Kulkarni, V., Chen, H., & Skiena, S. (2017, July). Don't Walk, Skip! Online learning of multi-scale network embeddings. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (pp. 258-265).

[16] Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500), 2323-2326.

[17] Shen, H., Cheng, X., Cai, K., & Hu, M. B. (2009). Detect overlapping and hierarchical community structure in networks. *Physica A: Statistical Mechanics and its Applications*, 388(8), 1706-1712.

[18] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015, May). LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 1067-1077).

[19] Wang, D., Cui, P., & Zhu, W. (2016, August). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1225-1234).

[20] Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., ... & Guo, M. (2018, April). GraphGAN: Graph representation learning with generative adversarial

nets. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).