Node2Vec-based Graph Embedding Neural Network Implementation for Shortest Path Prediction

CPS 3440 FA25 - 17 DEC 2025

Qiwen Xue 1306203
Chengyu Zhang 1308307
Hemu He 1307936
Ying Linjie 1306136
Kei Wai Hadassah Wong 1308584

CONTENTS

ABSTRACT

Our project is a learning-based shortest path prediction system that leverages Node2Vec graph representation learning with a regression neural network model.

Node2Vec was used to represent structural properties of graph nodes as low-dimensional vector embeddings.
A landmark-based graph node pair sampling approach was used to obtain graph node pairs with ground truth shortest path distances that can be utilized as supervisory signals for learning.
A multi-layer perceptron network was used to predict shortest path distances for each selected graph node pair by concatenating their vector embeddings.

We conducted experiments on synthetic social network graph datasets to show that the proposed prediction system can predict shortest path distances with small prediction errors, thereby establishing its feasibility as a rapid alternative to classical shortest path algorithms.

1

Introduction

INTRODUCTION

Background

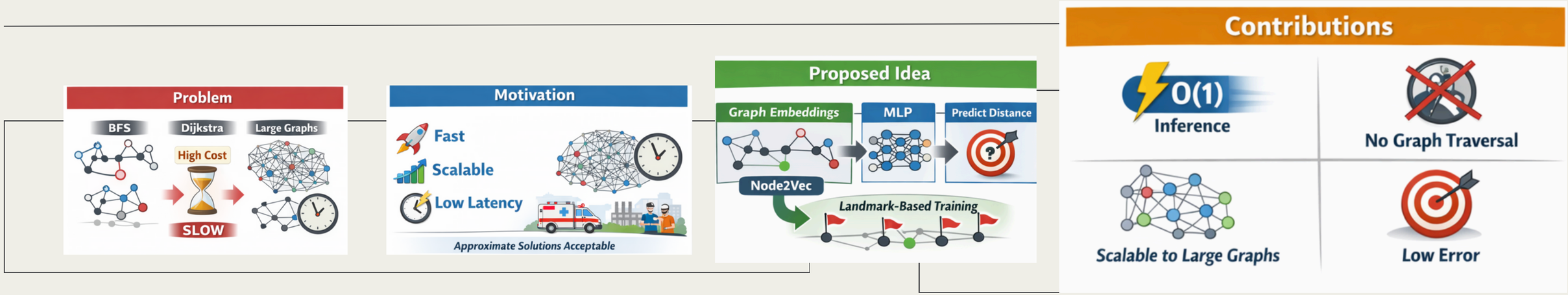
Graph-structured data is pervasive in a wide range of real-world applications, including social networking platforms, communication infrastructures, transportation systems, and biological interaction networks. These systems naturally model entities as nodes and their relationships as edges.

Many important graph-related tasks rely on understanding the distance relationships between nodes. Among them, the shortest path distance is one of the most fundamental and widely used metrics, serving as a basis for applications such as routing, network analysis, and node centrality computation.

In practical scenarios, shortest path information plays a critical role in decision-making processes, especially in time-sensitive and large-scale systems such as emergency response planning, traffic management, and network transmission optimization.

As real-world graphs continue to grow rapidly in size and complexity, efficient computation and querying of shortest path distances become increasingly challenging, motivating the exploration of scalable alternatives to traditional graph algorithms.

INTRODUCTION



Problem

Classical algorithms (BFS, Dijkstra) provide exact distances

Require graph traversal for every query

High computational cost for large-scale graphs and multiple queries

Unsuitable for low-latency and dynamic environments

Motivation

Need fast, scalable, and low-latency shortest path estimation

Approximate distances are acceptable in many real-time applications

Proposed Idea

Learn shortest path distances using graph embeddings + neural networks

Use Node2Vec to encode structural properties of nodes

Train an MLP to predict distances from node embedding pairs

Apply a landmark-based strategy for supervised training

Contributions

O(1) online inference complexity after offline training

No graph traversal during distance queries

Scalable to large graphs and query-intensive scenarios

Effective approximate shortest path prediction with low error

2

Related Work

RELATED WORK: SHORTEST PATHS

Traditional exact solution algorithms to shortest path problems like Dijkstra and Breadth-First Search (BFS) face limitations when computing large-scale graphs of millions of nodes.
However, approximate solutions such as the family of landmark-based algorithms prove to be better suited for handling such large graphs. Despite being able to handle large graphs, these algorithms trade perfect accuracy with significant improvements in time complexities.

- **Dijkstra's Algorithm**
- **Breadth-First Search (BFS)**
- **Landmark-based Algorithms**
- **Limitations/scalability issues**
- **Approximate solutions**
- **Landmark-based algorithms**
- **Trade-off: accuracy vs time complexity**
- **Computational efficiency**

RELATED WORK: GRAPH EMBEDDINGS

Graph embeddings transform nodes into low-dimensional vectors preserving structural properties. Traditional methods have quadratic complexity, impractical for large graphs.
Node2Vec offers linear time complexity using biased random walks.
Node2Vec uses parameters p (backtracking) and q (exploration vs exploitation) to capture both local and global structure. The implementation uses 128-dimensional embeddings, achieving 15% average error for shortest path prediction.

- **Node2Vec algorithm**
- **Biased random walks**
- **Linear time complexity O(n)**
- **Local and global structure**
- **Topological proximity preservation**
- **Scalable for large graphs**

3

Methodology

METHODOLOGY: 3 PHASES

1. **Graph Embedding Generation**
    - Use Node2Vec to create vector representations
2. **Training Data Construction**
    - Landmark-based sampling with Dijkstra
3. **Neural Network Training**
    - MLP learns mapping from embeddings to distances

METHODOLOGY: PHASE 1

EMBEDDING GENERATION

- Use Node2Vec algorithm to create vector representations for all nodes
- Each node → 128-dimensional vector
- Random walks capture both local and global graph structure
- Parameters: *p* and *q* control walk behavior (BFS-like vs DFS-like)
- Output: Embedding matrix where each node has a feature vector

METHODOLOGY: PHASE 2

TRAINING DATA CONSTRUCTION

- Select landmarks: Randomly choose 20 nodes as reference points
- Compute ground truth: Run Dijkstra's algorithm from each landmark to all other nodes
- Create training pairs: (landmark, node, actual_distance)
- Filter data:
- Remove self-loops (node == landmark)
- Exclude distance-1 pairs (direct neighbors)
- Optional: cap maximum distance
- Output: Dataset of node pairs with known shortest path distances

METHODOLOGY: PHASE 3

NEURAL NETWORK TRAINING

- **Input preparation: Concatenate embeddings of each node pair (256 dimensions total)**
- **Network architecture:**
  - **2 hidden layers (128 neurons each)**
  - **ReLU activation + Softplus output**
- **Training:**
  - **Optimizer: Adam (learning rate 0.05)**
  - **Loss function: Mean Squared Error (MSE)**
  - **30 epochs per graph, batch size 64**
- **Output: Trained model that predicts shortest path distance from any two node embeddings**

METHODOLOGY: ROLES

Dijkstra's Algorithm:

- Computes ground truth distances from landmarks to all nodes
- Creates training pairs (landmark, node, distance)
- Filters out self-loops and distance-1 pairs

Node2Vec Embeddings:

- Uses biased random walks controlled by parameters p and q
- *p* parameter: Controls backtracking (return probability)
- *q* parameter: Controls exploration vs exploitation (BFS-like vs DFS-like)
- Captures both local and global structure
- Chosen for flexibility and scalability

4

Experimental Evaluation

We first tried training the model on a social graph where all edge weights were set to 1.

The training results collapsed to a constant, meaning that regardless of the input, the output value remained the same.

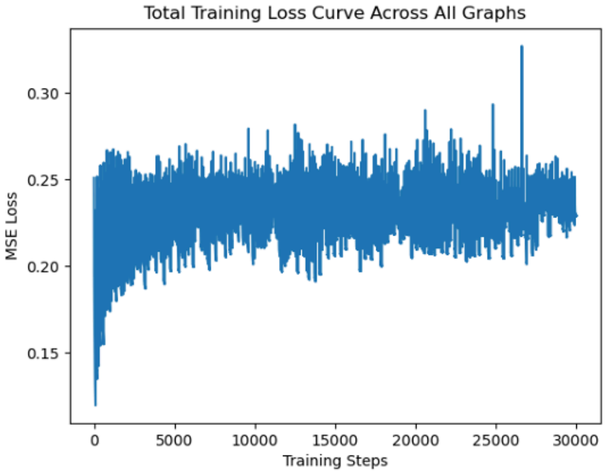


Fig. 2. Training MSE using BFS with unit edge weights.

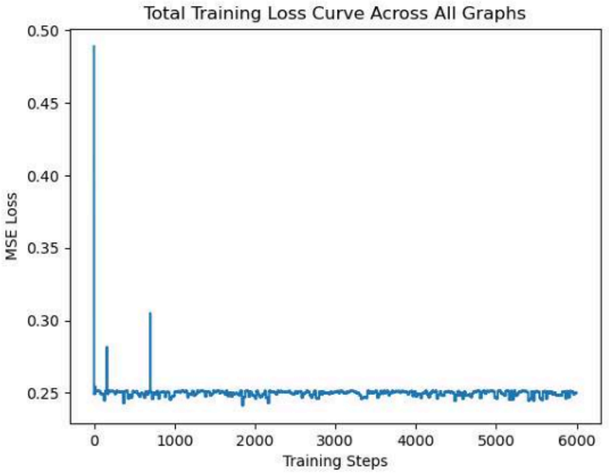


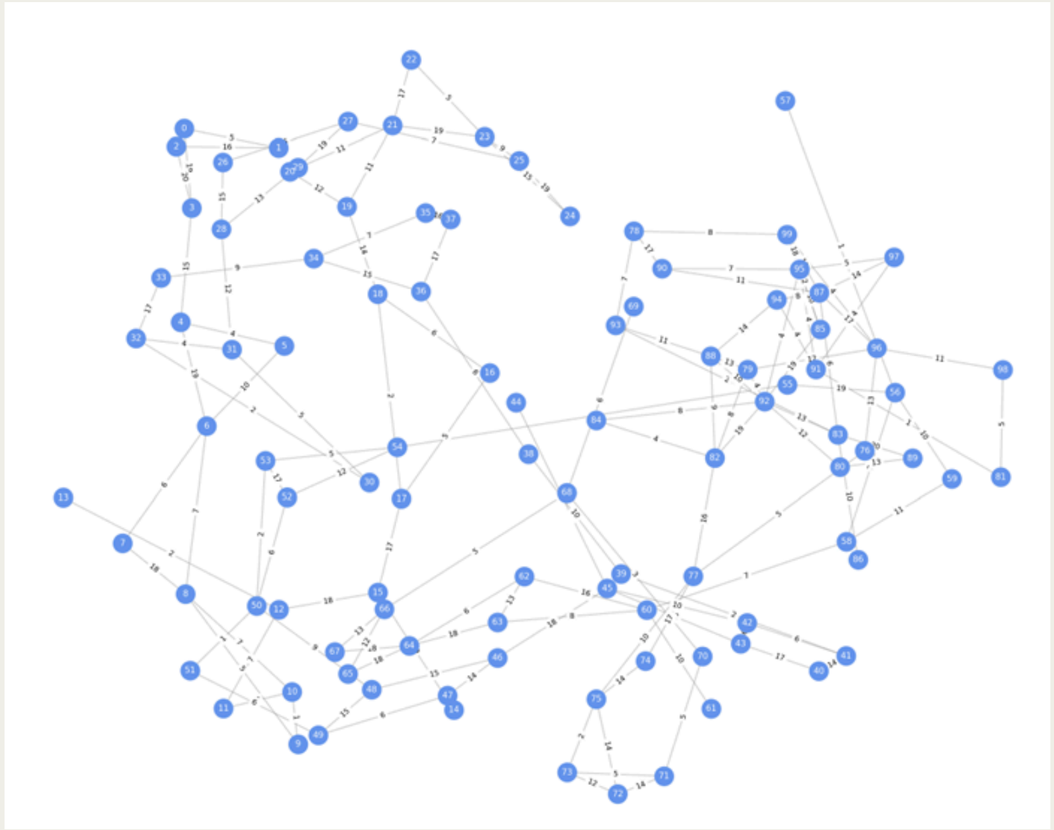
Fig. 3. Testing MSE using BFS with unit edge weights.

We identified two possible factors: relatively small edge values and an overly dense graph structure.

- **Small edge values：The distance values of node pairs were easily smoothed out during function approximation, making it difficult for the model to learn discriminative representations.**

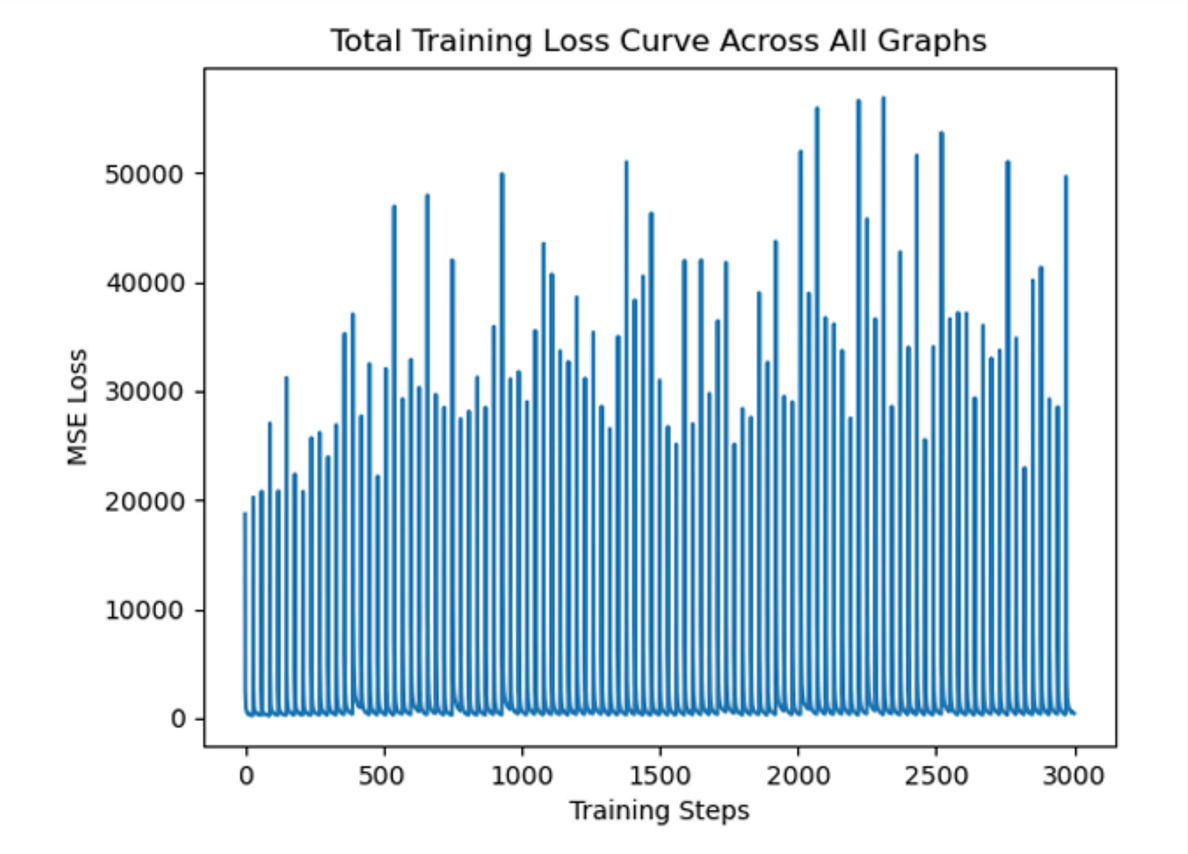
- **Overly dense graph structure：The limited variation in inter-node distances significantly reduced the structural diversity available for learning, thereby hindering the network’s ability to distinguish relationships among different users.**

We constructed our own graph for experimentation, and the graph we built was sparse.



- Number of layers: 30
- Number of edges between adjacent layers: 3
- Base number of random intra-layer edges: 2

100 nodes



The training results using Dijkstra's algorithm are shown in the figure.

5

Discussion

Embedding Dimension Selection and Performance Analysis

- **Key Finding:**

An embedding dimension of 128 achieves the optimal performance balance.

- **Detailed Analysis:**

1. Trade-off: Higher dimensions reduce error but significantly increase computational cost for embedding generation and distance calculation.
2. Why 128 is Optimal:
    - Effectiveness: node2vec's short random walks effectively capture local structural information.
    - Fitness: In scale-free networks where the average shortest path grows logarithmically, the distance itself is a local feature. A dimension of 128 successfully learns these node-pair distance features.
    - Diminishing Returns: Model expressiveness improves with higher dimensions, but the benefits gradually diminish beyond 128.

- **Implication:**

Dimension 128 represents the ideal trade-off point between prediction accuracy and computational overhead.

Error Distribution Across Different Path Lengths

- **Key Finding:**

An embedding dimension of 128 achieves the optimal performance balance.

- **Detailed Analysis:**
  - Two Contributing Factors:

    Data Imbalance: Insufficient long-distance samples in the training set lead to inadequate learning for long paths.

    Algorithmic Bias: node2vec's random walk strategy excels at capturing local neighborhood information but has limited capability for learning features of distant nodes.
  - Practical Implication:

    Our method is more suitable for distance prediction tasks involving short paths.

    This aligns perfectly with real-world scenarios (e.g., social networks), where most queries focus on shorter distances.

The Role of Binary Operators

- **Core Method:**

  Simple binary operators combine individual node embeddings to generate feature representations for node pairs.
- **Key Observation:**

  Operator performance is inconsistent across different datasets and dimensions.
- **Hypothesis:**

  This variation is likely related to the topological characteristics of the networks (e.g., strong community structure in Facebook vs. different connectivity in YouTube).
- **Future Research Directions:**
  - Why do operators perform differently across networks?
  - Are there more optimal operator combinations?
  - Which network topological features determine operator effectiveness?

6

Limitations & Future Works

FUTURE DIRECTIONS: DYNAMIC, HYBRID & GEOMETRIC

1. Adaptability to Dynamic Networks

- Current Limit: Static embedding requires full re-training.
- Proposed: Incremental Learning to update local subgraphs in real-time.

2. End-to-End Hybrid Architectures

- Current Limit: Decoupled pipeline (Node2Vec $\to$ MLP) causes info loss.
- Proposed: GNNs (GraphSAGE) for joint optimization of representation and prediction.

3. Integration of Hyperbolic Geometry

- Current Limit: Euclidean space distorts hierarchical structures (Scale-free).
- Proposed: Poincaré Embeddings to preserve tree-like hierarchy without distortion.

7

Conclusion

CONCLUSION

- **Performance Breakthrough**
  - Achieved O(1) constant-time queries (vs. BFS O(n+m)).
  - High accuracy preserved in local neighborhoods.

- **Methodological Innovation A: Inductive Learning**
  - Trained on independent graphs to Tested on unseen graphs.
  - Proved generalization capability (learning rules, not memorizing maps).

- **Methodological Innovation B: Synthetic Stress Testing**
  - Utilized Barabasi-Albert Model for controlled evaluation.
  - Validated robustness under varying network densities.

References{1}

[1] F. Salehi Rizi, J. Schloetterer, and M. Granitzer, “Shortest path distance approximation using deep learning techniques,” arXiv preprint. arXiv:2002.05257v1, 2020.

[2] X. Zhao, A. Sala, C. Wilson, H. Zheng, and B. Y. Zhao, “Orion: shortest path estimation for large social graphs,” Networks, vol. 1, p. 5, 2010.

[3] X. Zhao, A. Sala, H. Zheng, and B. Y. Zhao, “Efficient shortest paths on massive social graphs,” in 2011 7th International Conference on Collaborative Computing: Networking, Applications an Worksharing (CollaborateCom), pp. 77–86, IEEE, 2011.

[4] A. Gubichev, S. Bedathur, S. Seufert, and G. Weikum, “Fast and accurate estimation of shortest paths in large graphs,” in Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 499–508, ACM, 2010.

[5] J. B. Kruskal and M. Wish, Multidimensional Scaling, vol. 11. Sage, 1978.

[6] I. T. Jolliffe, “Principal component analysis and factor analysis,” Principal Component Analysis, pp. 115–128, 1986.

References {2}

[7] P. Goyal and E. Ferrara, “Graph embedding techniques, applications, and performance: A survey,” arXiv preprint arXiv:1705.02801, 2017.

[8] 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, pp. 701–710, ACM, 2014.

[9] 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, KDD ’16, (New York, NY, USA), pp. 855–864, ACM, 2016.

[10] L. Yang, C. Chatelain, and S. Adam, “Dynamic graph representation learning with neural networks: A survey,” IEEE Access, vol. 12, pp. 34233–34268, 2024.

[11] S. Bose, T. Anitha, N. Maheswaran, and D. Prabhu, “Adaptive deep learning techniques for real-time shortest path optimization in drone ambulance operations during disaster,” in Proceedings of the 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), IEEE, 2024.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms. Cambridge, Massachusetts: The MIT Press, 4th ed., 2022.