




Letter

SDGNN: Symmetry-Preserving Dual-Stream Graph Neural Networks

Jiufang Chen , Ye Yuan , and Xin Luo , Senior Member, IEEE

Dear Editor,

This letter proposes a symmetry-preserving dual-stream graph neural network (SDGNN) for precise representation learning to an undirected weighted graph (UWG). Although existing graph neural networks (GNNs) are influential instruments for representation learning to a UWG, they invariably adopt a unique node feature matrix for illustrating the sole node set of a UWG. Such a modeling strategy can limit the representation learning ability due to the diminished feature space. To this end, the proposed SDGNN innovatively adopts the following two-fold ideas: 1) Building a dual-stream graph learning framework that tolerates multiple node feature matrices for boosting the representation learning ability; 2) Integrating a symmetry regularization term into the learning objective for implying the equality constraint among its multiple node feature matrices, which exemplifies a graph's intrinsic symmetry and prompts learning the multiple node embeddings jointly. Experiments on six real-world UWG datasets indicate that the proposed SDGNN has superior performance in addressing the task of missing link estimation compared with the state-of-the-art baselines.

An UWG is the most commonly seen type of graph data from real application scenarios. Commonly, precise representation of a UWG is the foundation of subsequent pattern analysis like missing link estimation, anomaly detection, and cluster analysis, which has attracted widespread attention.

Related work: GNNs [1]–[5] are the linchpin constituent to the success of UWG representation learning. Given a UWG, a GNN model learns its low-dimensional node embeddings via the iterative aggregation of features describing its intrinsic neighborhoods. In recent years, many sophisticated GNN models with high performance have been proposed. For instance, Hamilton *et al.* [2] propose a GraphSAGE model that leverages the node feature information to efficiently generate node embeddings. Kong *et al.* [6] propose a novel mixed GCN model that utilizes a gate module to mix the linear and non-linear propagation information. Chen *et al.* [7] propose a residual GCN model that adopts the residual principle for high representation ability on a sparsely structured graph. He *et al.* [8] propose a LightGCN model that propagates the node embeddings linearly with the removal of feature transformations and nonlinear activations. Huang *et al.* [4] propose a GCN-RW model by modifying the convolutional layer with random filters and adjusting the learning objective using regularized least squares loss. Zou *et al.* [5] propose a spectral-based GCN for directed graphs that employs classical singular value decomposition to signal decomposition of the asymmetric adjacency matrix.

Despite the aforementioned GNN models' efficiency in addressing a UWG, they invariably adopt a unique node feature matrix for its representation. Such a diminished feature space inevitably restricts their representation learning ability. To break this bottleneck, this paper proposes a SDGNN with the following two-fold ideas: 1)

Corresponding authors: Ye Yuan and Xin Luo.

Citation: J. Chen, Y. Yuan, and X. Luo, "SDGNN: Symmetry-preserving dual-stream graph neural networks," *IEEE/CAA J. Autom. Sinica*, vol. 11, no. 7, pp. 1717–1719, Jul. 2024.

J. Chen is with the College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, and the Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China (e-mail: d210201001@stu.cqupt.edu.cn).

Y. Yuan and X. Luo are with the College of Computer and Information Science, Southwest University, Chongqing 400715, China (e-mail: yuanyekl@swu.edu.cn; luoxin@swu.edu.cn).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/JAS.2024.124410

Building a dual-stream graph learning framework that tolerates multiple node embeddings to boost the representation learning ability; 2) Introducing a symmetry regularization term into the learning objective for elegantly imply the equality constraint among its multiple node feature matrices, which can exemplify a graph's intrinsic symmetry and prompts to learning the multiple node embeddings jointly.

Problem statement: Given a UWG $G = (V, E)$ as V denotes a set of n nodes and E represents a set of m edges, it can be expressed by a symmetric adjacency matrix $A = [a_{i,c}]^{n \times n}$ where $a_{i,c}$ is the non-zero weight if $e_{i,c} \in E$ and zero otherwise. For efficient representation learning to G , each node can be described by a feature vector $x_i \in \mathbb{R}^f$, where $X \in \mathbb{R}^{n \times f}$ is the f -dimensional feature matrix and $f \ll n$.

In particular, a GNN model learns the node embedding by propagating its information of neighbors as follows:

$$z_i^{(l)} = \text{AGGFUN}(z_j^{(l-1)} | j \in N(i)) \quad (1)$$

where $N(i)$ is the neighbor node set of node v_i , l is the number of GNN layers, z_i is the embedding of node v_i , and $\text{AGGFUN}(\cdot)$ is the aggregation function defined by different GNN modeling strategies such as GCN [9], GAT [10] and LightGCN [8]. Since a GNN model sets the initial node embedding as the node features, i.e., $z_i^0 = x_i$, the final node embedding for nodes v_i and v_c are z_i and z_c after L -layers' propagation. As indicated by [11], [12], the estimation for the link weight between two nodes can be represented by $\hat{a}_{i,c} = \langle z_i, z_c \rangle$ as $\langle \cdot, \cdot \rangle$ denotes the inner product between two vectors.

The proposed SDGNN model: Now we introduce the proposed SDGNN model, whose architecture is shown in Fig. 1. Note that it is model-agnostic, i.e., it can be compatible with most existing GNN variants, i.e., GCN, GAT, and LightGCN. Hence, in our context, we choose the LightGCN model as the backbone to introduce the proposed SDGNN due to its convenience of implementation [8].

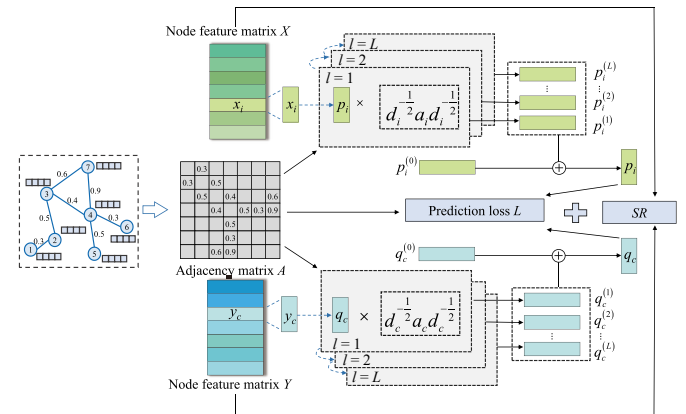


Fig. 1. A framework of SDGNN.

Propagation and layer combination: Note that the proposed SDGNN contains a dual-stream graph learning structure. Hence, we randomly initialize two independent feature matrices $X \in \mathbb{R}^{n \times f}$ and $Y \in \mathbb{R}^{n \times f}$ for each stream. In LightGCN, the feature transformation and nonlinear activation are neglected. Thus, the node embedding is

$$p_i^{(l)} = \sum_{j \in N(i)} \frac{1}{|N(i)|} p_j^{(l-1)}, \quad q_c^{(l)} = \sum_{d \in N(c)} \frac{1}{|N(c)|} q_d^{(l-1)} \quad (2)$$

where p_i and q_c denote the embedding of nodes v_i and v_c in the first and second stream, $N(i)$ and $N(c)$ refer to the directly connected neighbor set of nodes v_i and v_c , respectively. Note that we have $p_i^{(0)} = x_i$ and $q_c^{(0)} = y_c$ denote the i -th and c -th row vectors corresponding to X and Y .

Based on the layer combination rule of LightGCN, the node embedding after L -layers GNN propagation is given as

$$p_i = \sum_{l=0}^L \alpha_l p_i^{(l)}, \quad q_c = \sum_{l=0}^L \alpha_l q_c^{(l)} \quad (3)$$

where $\alpha_l \geq 0$ denotes the coefficient to balance the importance of each layer and can be set uniformly as $1/(L+1)$. Since this paper

adopts the missing link estimation as the downstream task to verify the representation ability of SDGNN, the unknown link weight based on the obtained node embedding is estimated as

$$\hat{a}_{i,c} = \langle p_i, q_c \rangle \quad (4)$$

where $\hat{a}_{i,c}$ is the estimated missing link between node v_i and v_c .

Symmetry preserving: Note that the adjacency matrix A of a UWG is symmetric, i.e., $a_{i,c} = a_{c,i}$ for nodes v_i and v_c . Ideally, the estimated adjacency matrix \hat{A} should be also symmetric, i.e., $\hat{a}_{i,c} = \hat{a}_{c,i}$. However, the dual-stream graph learning structure fails to preserve the symmetry of \hat{A} because we commonly have $\hat{a}_{i,c} = \langle p_i, q_c \rangle \neq \hat{a}_{c,i} = \langle p_c, q_i \rangle$. On the other hand, symmetry is the intrinsic characteristic of a UWG and should be treated carefully for building its precise representation. To address this issue, this paper builds the equality constraint between the dual embeddings for the same node. In detail, for $\forall v_i \in V$, it is equivalent to minimizing the equation

$$\|p_i - q_i\|^2 \rightarrow 0. \quad (5)$$

Based on (2) and (3), the following deduction is obtained:

$$p_i = \alpha_0 p_i^{(0)} + \alpha_1 \sum_{j \in N(i)} \frac{1}{|N(i)|} p_j^{(0)} + \dots + \alpha_l \sum_{j \in N(i)} \frac{1}{|N(i)|^L} p_j^{(0)}. \quad (6)$$

Similarly, the expression of q_i under the same situation is given as

$$q_i = \alpha_0 q_i^{(0)} + \alpha_1 \sum_{j \in N(i)} \frac{1}{|N(i)|} q_j^{(0)} + \dots + \alpha_l \sum_{j \in N(i)} \frac{1}{|N(i)|^L} q_j^{(0)}. \quad (7)$$

From (6) and (7), we clearly see that the node embedding of the same node v_i in different streams is a linear combination of their related node feature. Based on (5)–(7), we have

$$\left\| \begin{aligned} & \left(\alpha_0 p_i^{(0)} + \alpha_1 \sum_{j \in N(i)} \frac{1}{|N(i)|} p_j^{(0)} + \dots + \alpha_l \sum_{j \in N(i)} \frac{1}{|N(i)|^L} p_j^{(0)} \right) \\ & - \left(\alpha_0 q_i^{(0)} + \alpha_1 \sum_{j \in N(i)} \frac{1}{|N(i)|} q_j^{(0)} + \dots + \alpha_l \sum_{j \in N(i)} \frac{1}{|N(i)|^L} q_j^{(0)} \right) \end{aligned} \right\|^2 \rightarrow 0. \quad (8)$$

Note that the initial node embedding is set to the node features. Hence, for $\forall v_i \in V$, (8) is equivalent to

$$\|p_i^{(0)} - q_i^{(0)}\|^2 = \|x_i - y_i\|^2 \rightarrow 0. \quad (9)$$

Based on the above analyses, a symmetry regularization (SR) term is obtained for minimizing the distance between X and Y

$$SR = \|X - Y\|^2 \quad (10)$$

where $\|\cdot\|^2$ denotes the calculation of L_2 norm. With it, the intrinsic symmetry of a UWG can be preserved elegantly.

Model training: Since the feature transformation is abandoned, the only optimization parameter of the proposed SDGNN model is the node embeddings of the 0th layer, i.e., the node feature matrices X and Y . Hence, we build the following learning objective to jointly train the dual node embeddings with the incorporation of the SR term:

$$\varepsilon = L + \frac{\kappa}{2} SR = \sum_{a_{i,c} \in \Lambda} (a_{i,c} - \hat{a}_{i,c})^2 + \frac{\kappa}{2} \|X - Y\|^2 \quad (11)$$

where Λ denotes the known weight set, κ denotes the regularization coefficient of the SR term, respectively. We adopt a mini-batch Adam algorithm for training the optimization parameters.

Algorithm design and analysis: The algorithm of SDGNN is summarized in Algorithm 1. Its computation is given as: 1) Initialization: $T_{S1} = \Theta(2 \times |N| \times f)$; and 2) Parameter training: $T_{S2} = \Theta(|\Lambda| \times f \times L \times \lceil |N|/\text{batch size} \rceil \times t)$, which sum up to

$$T = T_{S1} + T_{S2} \approx \Theta \left(|\Lambda| \times f \times L \times \left\lceil \frac{|N|}{\text{batch size}} \right\rceil \times t \right). \quad (12)$$

On the other hand, its storage cost mainly depends on its modules $S1$ and $S2$, which sum up to

$$S = S_{S1} + S_{S2} = \Theta(2|N| \times f) + \Theta(\Lambda) + \Theta(\Lambda) = 2\Theta(\Lambda + |N| \times f). \quad (13)$$

Experiments: The experiments were performed on a platform with a 2.4 GHz Intel Xeon 4214R CPU and four NVIDIA RTX 3090 GPUs with 128 GB RAM. Six UWG datasets are adopted in Table 1. The target is to testify the estimation accuracy for missing link weight by the involved models, which can be quantified by the root mean squared error (RMSE) and mean absolute error (MAE) [13]

$$\text{RMSE} = \sqrt{\left\| \left(\sum_{a_{i,c} \in \Gamma} (a_{i,c} - \hat{a}_{i,c})^2 \right) \right\| / |\Gamma|}, \text{MAE} = \left(\sum_{a_{i,c} \in \Gamma} |a_{i,c} - \hat{a}_{i,c}|_{\text{abs}} \right) / |\Gamma|.$$

Table 1. Details of Adopted Graph

| No. | Networks | Nodes | Edges | Density |
|-----|---------------------|--------|---------|---------|
| D1 | Collins [14] | 1 000 | 8 246 | 1.65% |
| D2 | Plants [14] | 1 600 | 10 965 | 0.86% |
| D3 | Plantsmargin [14] | 1 600 | 12 741 | 1.00% |
| D4 | Yeast [14] | 1 484 | 31 175 | 2.83% |
| D5 | JapaneseVowels [14] | 9 961 | 65 572 | 0.13% |
| D6 | Worms20 [14] | 20 055 | 120 413 | 0.06% |

Algorithm SDGNN

```

Input:  $\Lambda$ 
/*Initialization*/
1: Initialize: node feature matrix  $X$  and  $Y$ , propagation layer  $L = 3$ . S1
   Initialize:  $\eta, \kappa$ , batch size, iteration count  $t = 0$  and max-iteration-count  $T$ .
/*Learning Operation*/
2: Calculate  $\tilde{A}$  according to  $\tilde{A} = D^{-0.5} \tilde{A} D^{-0.5}$ ;
3: while not converging and  $t \leq T$  do
4:   for batch = 1 to  $\lceil |N|/\text{batch size} \rceil$ 
5:     for  $l = 1$  to  $L$ 
6:       Update  $P^{(l)}$  and  $Q^{(l)}$  according to (2);
7:       Calculate  $P$  and  $Q$  according to (3);
8:       Sample a batch size  $\times |N| \Lambda_{\text{batch}}$  form  $\Lambda$ ;
9:       for  $s = 1$  to batch size S2
10:        for  $n \in \Lambda(s)$ 
11:          Calculate  $\hat{a}_{i,c}$  according to (4);
12:          Calculate the loss using (11);
13:          Accumulate the losses in  $\Lambda_{\text{batch}}$ ;
14:          Calculate the gradients through backpropagation;
15:          Update all the variables according to the gradients;
/*Operation Ending*/
Output:  $X$  and  $Y$ 

```

We compare the proposed SDGNN with several state-of-the-art and related models: MF [13], NeuMF [15], GCN [1], GC-MC [16], LR-GCCF [7], LightGCN [8], DGCN HN [17], SGL_ED [18], HMLLET [6], LightGCN_{r-AdjNorm} [19] and GDE [20]. To ensure fair comparisons, all models adopt the Adam optimizer with a fixed batch size of 2048. We conduct a grid search to tune the optimal results, i.e., the learning rate η is tuned amongst $\{5E-5, 1E-4, 5E-4, 1E-3, 5E-3, 0.01, 0.05\}$ and the regularization coefficient κ is amongst $\{1E-5, 5E-5, 1E-4, 5E-4, 1E-3, 5E-3, 0.01, 0.05, 0.1, 0.5, 1\}$. The node feature dimension is $f = 128$ and the propagation layer of SDGNN is fixed at $L = 3$ according to [8].

Comparison results: All models were repeatedly trained ten times with random initializations to achieve the final averaged results. Table 2 presents the average RMSE and MAE of compared models, along with statistical results, i.e., win/loss counts, Friedman rank, and Wilcoxon test p -values. We have the following findings.

1) SDGNN obtains a more accurate representation learning to a UWG than its peers do. As shown in Table 2, it significantly outperforms the rival models on eleven testing cases out of twelve in total. However, the situation is different on D1, since SDGNN is outperformed by NeuMF in RMSE. Kindly note that D1 is the smallest dataset in our experiments, where the risk of overfitting for SDGNN is the largest. In general, SDGNN's performance gain has statistical significance across the experiments according to the Friedman rank and p -value results.

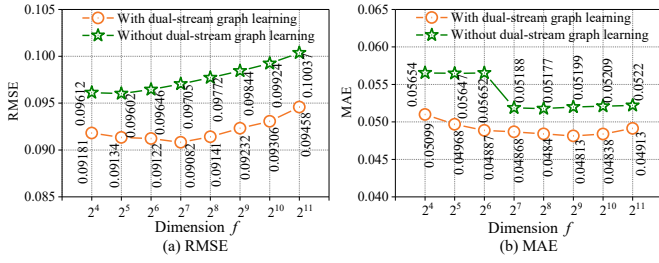
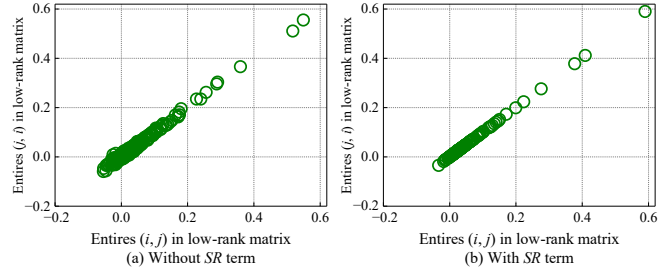
2) SDGNN utilizes two node feature matrices instead of increasing the node feature dimension to break through the limit of representation learning ability. As shown in Fig. 2, when implementing a dual-stream graph learning framework, the RMSE and MAE are always lower than the one with a unique node feature matrix. Specifically, the SDGNN model with the dual-stream settings still outperforms the model without dual-stream even if its feature dimension is set at $2 \times f$. For instance, on D2, the RMSE with the dual-stream is 0.090 82 with $f = 128$, while the RMSE without the dual-stream is 0.097 72 with $f = 256$, as shown in Fig. 2(a). The main reason is that the representation ability does not increase continuously as the node feature dimension increases, as shown in Fig. 2(a). Hence, the strong learning ability of SDGNN brought by its dual-stream is vital for its high representation accuracy.

Symmetry verification: Fig. 3 shows the data distribution of score estimation on D2 w/o the SR term. From it, we find that only partial symmetry of the target UWG is captured without the SR term. In contrast, as depicted in Fig. 3(b), SDGNN with the SR term enables the

Table 2. The Comparison Results on RMSE/MAE(\pm std), Where \star Shows That SDGNN is Outperformed by its Peers

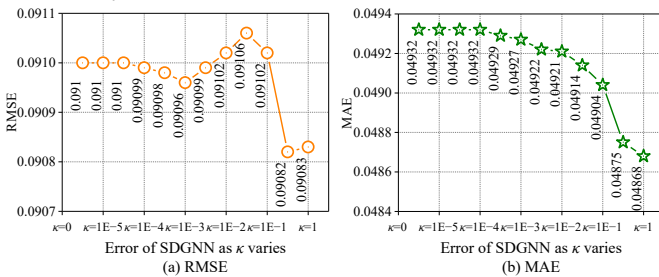
| No. | Metric | MF | NeuMF | GCN | GC-MC | LR-GCCF | LightGCN | DGCN_HN | SGL_ED | HMLET | LightGCN _{AdjNorm} | GDE | SDGNN |
|-----------|--------|--------------------------|----------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-----------------------------|--------------------------|----------------------------------|
| D1 | RMSE | 0.13821 _{±2E-4} | \star 0.11112 _{±7E-4} | 0.12478 _{±3E-3} | 0.11808 _{±4E-3} | 0.11774 _{±1E-3} | 0.13008 _{±9E-5} | 0.14507 _{±1E-4} | 0.15465 _{±6E-4} | 0.14364 _{±1E-3} | 0.13754 _{±8E-5} | 0.16916 _{±2E-3} | 0.11262 _{±2E-4} |
| | MAE | 0.08419 _{±1E-4} | 0.07849 _{±4E-4} | 0.08506 _{±1E-3} | 0.08240 _{±3E-3} | 0.08242 _{±8E-4} | 0.08393 _{±1E-4} | 0.09587 _{±5E-5} | 0.09991 _{±2E-4} | 0.09054 _{±1E-3} | 0.08444 _{±5E-5} | 0.12407 _{±6E-3} | \star 0.07157 _{±3E-4} |
| D2 | RMSE | 0.10245 _{±9E-5} | 0.09569 _{±9E-4} | 0.09630 _{±1E-3} | 0.09889 _{±1E-3} | 0.09343 _{±2E-4} | 0.09705 _{±5E-5} | 0.10192 _{±4E-5} | 0.10346 _{±4E-4} | 0.10049 _{±2E-4} | 0.10224 _{±2E-4} | 0.10677 _{±3E-4} | \star 0.09082 _{±1E-4} |
| | MAE | 0.05343 _{±5E-5} | 0.05479 _{±8E-4} | 0.05479 _{±1E-3} | 0.05792 _{±1E-3} | 0.05308 _{±3E-4} | 0.05188 _{±3E-4} | 0.05315 _{±8E-5} | 0.05494 _{±4E-4} | 0.05172 _{±2E-4} | 0.05331 _{±1E-4} | 0.06088 _{±7E-4} | \star 0.04868 _{±1E-4} |
| D3 | RMSE | 0.04113 _{±7E-5} | 0.03756 _{±3E-4} | 0.03844 _{±7E-5} | 0.03866 _{±5E-4} | 0.03920 _{±2E-4} | 0.03886 _{±3E-4} | 0.03911 _{±8E-5} | 0.04300 _{±6E-4} | 0.03974 _{±1E-4} | 0.04085 _{±1E-4} | 0.04399 _{±2E-4} | \star 0.03658 _{±7E-5} |
| | MAE | 0.02301 _{±2E-5} | 0.02270 _{±3E-4} | 0.02285 _{±2E-4} | 0.02291 _{±3E-4} | 0.02272 _{±8E-5} | 0.02213 _{±3E-5} | 0.02218 _{±3E-5} | 0.02627 _{±2E-4} | 0.02179 _{±5E-5} | 0.02278 _{±4E-5} | 0.02633 _{±2E-4} | \star 0.02117 _{±7E-5} |
| D4 | RMSE | 0.08806 _{±7E-5} | 0.08540 _{±2E-4} | 0.08585 _{±6E-4} | 0.07973 _{±2E-3} | 0.08590 _{±1E-3} | 0.08132 _{±7E-5} | 0.08009 _{±9E-5} | 0.09020 _{±4E-4} | 0.08379 _{±3E-4} | 0.08717 _{±9E-5} | 0.09383 _{±6E-4} | \star 0.07131 _{±1E-4} |
| | MAE | 0.04519 _{±2E-5} | 0.05378 _{±3E-4} | 0.05208 _{±4E-4} | 0.05010 _{±1E-3} | 0.04888 _{±2E-4} | 0.04375 _{±8E-5} | 0.04177 _{±3E-5} | 0.04687 _{±7E-5} | 0.04372 _{±3E-5} | 0.04668 _{±9E-5} | 0.05999 _{±3E-4} | \star 0.04061 _{±1E-4} |
| D5 | RMSE | 0.12670 _{±3E-4} | 0.11887 _{±4E-4} | 0.11943 _{±7E-4} | 0.11926 _{±2E-4} | 0.11637 _{±2E-5} | 0.11921 _{±4E-5} | 0.11884 _{±3E-5} | 0.12005 _{±4E-5} | 0.12062 _{±1E-4} | 0.12379 _{±8E-5} | 0.12269 _{±5E-5} | \star 0.11467 _{±3E-5} |
| | MAE | 0.06237 _{±4E-5} | 0.05939 _{±8E-4} | 0.06566 _{±4E-5} | 0.06759 _{±1E-3} | 0.06364 _{±3E-4} | 0.05995 _{±3E-4} | 0.05929 _{±5E-5} | 0.05882 _{±3E-5} | 0.05912 _{±1E-4} | 0.05958 _{±9E-5} | 0.06418 _{±2E-4} | \star 0.05802 _{±4E-5} |
| D6 | RMSE | 0.22869 _{±2E-4} | 0.19475 _{±2E-3} | 0.19707 | 0.20195 | 0.19382 _{±3E-4} | 0.21161 _{±5E-5} | 0.22547 _{±8E-5} | 0.22616 _{±7E-4} | 0.22625 _{±2E-4} | 0.22550 _{±1E-4} | 0.23305 _{±4E-4} | \star 0.18877 _{±2E-4} |
| | MAE | 0.13741 _{±4E-4} | 0.13981 _{±2E-3} | 0.12838 _{±1E-3} | 0.14098 _{±8E-4} | 0.13464 _{±1E-4} | 0.13375 _{±5E-5} | 0.14593 _{±7E-5} | 0.14059 _{±1E-4} | 0.13937 _{±2E-4} | 0.13830 _{±2E-4} | 0.15952 _{±2E-3} | \star 0.11653 _{±6E-5} |
| Win/Loss | — | 12/0 | 11/1 | 12/0 | 12/0 | 12/0 | 12/0 | 12/0 | 12/0 | 12/0 | 12/0 | 12/0 | — |
| Rank* | — | 8.50 | 5.00 | 6.42 | 6.75 | 4.92 | 4.67 | 6.08 | 9.17 | 6.17 | 7.58 | 11.67 | \star 1.08 |
| p-value** | — | 2.44E-4 | 1.22E-3 | 2.44E-4 | 2.44E-4 | 2.44E-4 | 2.44E-4 | 2.44E-4 | 2.44E-4 | 2.44E-4 | 2.44E-4 | 2.44E-4 | — |

*A lower Friedman rank value indicates a higher accuracy. **The accepted hypotheses with a significance level of 0.05 are highlighted.

Fig. 2. Error with/without dual-stream graph learning on D2 as f varies.Fig. 3. Data distribution in \hat{a} on D2.

symmetry representation more precisely.

Effect of parameters: Fig. 4 records the SDGNN’s performance as κ values. From it, we evidently see that RMSE and MAE change significantly as κ varies. Specifically, MAE decreases as κ increases continuously. Hence, κ should be tuned with care.

Fig. 4. Errors of SDGNN on D2 as κ varies.

Conclusions: This letter proposes the SDGNN model to implement the precise representation of a UWG. It achieves significant improvements in missing link estimation within a UWG than state-of-the-art models do. We plan to investigate the theoretical evidence regarding the reason why the proposed dual-stream graph learning structure can break the limit of representation learning ability next.

Acknowledgments: This work was supported in part by the National Natural Science Foundation of China (62372385, 62002337) and the Chongqing Natural Science Foundation (CSTB2022NSCQ-MSX1486, CSTB2023NSCQ-LZX0069).

References

- [1] M. Welling and T. N. Kipf, “Semi-supervised classification with graph convolutional networks,” arXiv preprint arXiv: 1609.02907, 2016.
- [2] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Proc. Adv. Neural Inf. Proc. Syst.*, 2017, pp. 1024–1034.
- [3] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.
- [4] C. Huang, M. Li, F. Cao, *et al.*, “Are graph convolutional networks with random weights feasible?” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 2751–2768, 2023.
- [5] C. Zou, A. Han, L. Lin, M. Li, and J. Gao, “A simple yet effective framelet-based graph neural network for directed graphs,” *IEEE Trans. Artif. Intell.*, vol. 5, no. 4, pp. 1647–1657, 2024.
- [6] T. Kong, T. Kim, J. S. Jeon, J. W. Choi, Y. C. Lee, N. Park, and S. W. Kim, “Linear, or non-linear, that is the question!” in *Proc. 15th ACM Int. Conf. Web Search and Data Min.*, 2022, pp. 517–525.
- [7] L. Chen, L. Wu, R. Hong, *et al.*, “Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach,” in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 27–34.
- [8] X. He, K. Deng, X. Wang, *et al.*, “LightGCN: Simplifying and powering graph convolution network for recommendation,” in *Proc. 43rd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, 2020, pp. 639–648.
- [9] X. Liu, M. Yan, L. Deng, *et al.*, “Sampling methods for efficient training of graph convolutional networks: A survey,” *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 2, pp. 205–234, 2022.
- [10] T. He, S. O. Yew, and B. Lu, “Learning conjoint attentions for graph neural nets,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 2641–2653.
- [11] Z. G. Liu, X. Luo, and M. Zhou, “Symmetry and graph bi-regularized non-negative matrix factorization for precise community detection,” *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 2, pp. 1406–1420, 2024.
- [12] F. Bi, T. He, Y. Xie, and X. Luo, “Two-stream graph convolutional network-incorporated latent feature analysis,” *IEEE Trans. Serv. Comput.*, vol. 16, no. 4, pp. 3027–3042, 2023.
- [13] X. Luo, D. Wang, M. Zhou, and H. Yuan, “Latent factor-based recommenders relying on extended stochastic gradient descent algorithms,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 51, no. 2, pp. 916–926, 2021.
- [14] T. A. Davis and Y. F. Hu, “The university of Florida sparse matrix collection,” *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1–25, 2011.
- [15] X. He, L. Liao, H. Zhang, *et al.*, “Neural collaborative filtering,” in *Proc. Int. Conf. World Wide Web*, 2017, pp. 173–182.
- [16] R. V. Berg, T. N. Kipf, and M. Welling, “Graph convolutional matrix completion,” arXiv preprint arXiv: 1706.02263, 2017.
- [17] W. Guo, Y. Yang, Y. Hu, *et al.*, “Deep graph convolutional networks with hybrid normalization for accurate and diverse recommendation,” in *Proc. Workshop Deep Learning Practice for High-Dimensional Sparse Data With KDD*, 2021.
- [18] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, “Self-supervised graph learning for recommendation,” in *Proc. 44th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, 2021, pp. 726–735.
- [19] M. Zhao, L. Wu, Y. Liang, *et al.*, “Investigating accuracy-novelty performance for graph-based collaborative filtering,” in *Proc. 45th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, 2022, pp. 50–59.
- [20] S. Peng, K. Sugiyama, and T. Mine, “Less is more: Reweighting important spectral graph features for recommendation,” in *Proc. 45th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, 2022, pp. 1273–1282.