Scalable and Effective Bipartite Network Embedding

Renchi Yang
renchi@nus.edu.sg
National University of Singapore

Jieming Shi
Hong Kong Polytechnic University
jieming.shi@polyu.edu.hk

Keke Huang
National University of Singapore
kkhuang@nus.edu.sg

Xiaokui Xiao
National University of Singapore
xkxiao@nus.edu.sg

ABSTRACT

Given a bipartite graph G consisting of inter-set weighted edges connecting the nodes in two disjoint sets U and V, *bipartite network embedding* (BNE) maps each node $u_i \in U$ and $v_j \in V$ to compact embedding vectors that capture the hidden topological features surrounding the nodes, to facilitate downstream tasks. Effective BNE should preserve not only the direct connections between nodes but also the multi-hop relationships formed alternately by the two types of nodes in G, which can incur prohibitive overheads, especially on massive bipartite graphs with millions of nodes and billions of edges. Existing solutions are hardly scalable to massive bipartite graphs, and often produce low-quality results.

This paper proposes GEBE, a generic BNE framework achieving state-of-the-art performance on massive bipartite graphs, via four main algorithmic designs. First, we present two generic measures to capture the multi-hop similarity/proximity between homogeneous/heterogeneous nodes respectively, and the measures can be instantiated with three popular probability distributions, including Poisson, Geometric, and Uniform distributions. Second, GEBE formulates a novel and unified BNE objective to preserve the two measures of all possible node pairs. Third, GEBE includes several efficiency designs to get high-quality embeddings on massive graphs. Finally, we observe that GEBE achieves the best performance when instantiating MHS and MHP using a Poisson distribution, and thus, we further develop GEBE^p based on Poisson-instantiated MHS and MHP, with non-trivial efficiency optimizations. Extensive experiments, comparing 15 competitors on 10 real datasets, demonstrate that our solutions, especially GEBE^p, obtain superior result utility than all competitors for top-N recommendation and link prediction, while being up to orders of magnitude faster.

CCS CONCEPTS

• **Mathematics of computing** → *Graph algorithms*; • **Computing methodologies** → *Learning latent representations*; *Factorization methods*; *Unsupervised learning*.

KEYWORDS

Bipartite Graphs; Network Embedding; Poisson Distribution

ACM Reference Format:
Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. 2022. Scalable and Effective Bipartite Network Embedding. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3514221.3517838

1 INTRODUCTION

Bipartite graphs are ubiquitous with numerous applications, such as movie recommendation on user-movie rating sites [4], product recommendation on online shopping platforms [21, 41], drug discovery using drug-target interaction data [71], query rewriting based on query-ad click logs [2], and author-publication correlation inference [81]. Bipartite network embedding (BNE) is a fundamental task to support these applications, and has attracted much attention from academia [7, 24] and industry [27, 76]. High-quality BNE can facilitate accurate predictions in downstream machine learning tasks, *e.g.*, top-N recommendation [65, 77] and link prediction [7, 24].

A bipartite graph has two types of nodes, and each node could influence not only its direct neighbors, but also other nodes that could be reached via multi-hop paths. Naturally, an effective BNE solution should capture both the *multi-hop similarity* between same-typed nodes and the *multi-hop proximity* between nodes of different types. Although there exist a number of measures to evaluate the similarity/proximity between nodes in a graph (*e.g.*, Personalized PageRank (PPR) [34], heat kernel PageRank (HKPR) [12], and uniform high-order proximity [72]), they are not tailored for bipartite graphs, and hence, are not optimized for BNE.

In addition, existing BNE solutions incur significant computational overheads, and have difficulties scaling to massive graphs with millions of nodes and billions of edges. In particular, as shown in Section 6, the two latest BNE methods, BiNE [24] and BiGI [7], are unable to return embedding results within three days on a dataset with 1.1 billion edges. Specifically, BiNE requires sampling a large number of random walks to estimate the similarity/proximity between nodes, which is prohibitively expensive. Meanwhile, BiGI requires using multilayer perceptrons (MLPs) [54] to train embeddings, leading to substantial computational costs.

One may also attempt to treat bipartite graphs as general graphs, and then apply general network embedding methods (*e.g.*, [18, 26, 50, 57, 60, 61, 69, 74, 75, 80, 84, 85, 88]) for BNE computation. This approach, however, yields sub-optimal results, since general network embedding methods overlook the bipartite structures [7].

To remedy the deficiency of existing methods, we propose GEBE, a GEneric Bipartite Network Embedding framework that significantly advances the state of the art in BNE computation, in terms of both result quality and efficiency. We first propose two *generic* measures, namely, multi-hop homogeneous similarity (MHS) for nodes of the same type and multi-hop heterogeneous proximity (MHP) for nodes of different types, to evaluate the relationships between nodes. MHS and MHP can be used to encapsulate various probability distributions used in the literature, including *Geometric distribution* used in PPR [34], *Poisson distribution* used in HKPR [12], and *Uniform distribution* applied in uniform high-order proximity [72]. Then we formulate the BNE task as an optimization problem with a unified objective of approximating both MHS and MHP of all possible node pairs as a whole. Directly achieving this objective is non-trivial, since it requires the materialization of an $O(|U| \cdot |V|)$-sized matrix and an $O(|U|^2)$-sized matrix. To circumvent this, we convert the optimization task into the computation of an eigen-decomposition, and we propose several techniques in GEBE to efficiently derive embeddings based on the decomposition.

In addition, we observe that when we instantiate MHS and MHP using a Poisson distribution, the embeddings derived from GEBE offer superior performance for top-N recommendation and link prediction, as shown in Section 6. Motivated by this, we further develop GEBE^p based on Poisson-instantiated MHS and MHP, with non-trivial efficiency optimizations that exploit the properties of Poisson distributions. We prove that GEBE^p runs in time almost linear to the bipartite graph size.

We conduct extensive experiments using 10 real datasets and comparing against 15 existing methods. Our experimental results demonstrate that the proposed methods, especially GEBE^p, consistently generate high-quality embeddings with superior predictive accuracy for top-N recommendation and link prediction, while being up to orders of magnitude faster. For instance, on our largest dataset with 1.1 billion edges, GEBE^p terminates within 2 hours using a single thread on a commodity machine, and achieves 26.5% F1 score for top-10 recommendation and 95.8% Precision-Recall for link prediction, while most competitors are unable to finish BNE computation in three days.

To sum up, we make the following contributions in our paper.

- We introduce two generic measures, MHS for homogeneous nodes and MHP for heterogeneous nodes, to evaluate the multi-hop relationships between nodes in a bipartite graph.
- We propose GEBE, a generic BNE framework with a unified objective to preserve the MHS and MHP of all possible node pairs in a bipartite graph. GEBE is general to support three instantiations using Geometric, Poisson, and Uniform distributions.
- For the purpose of efficiency, we convert the optimization of the objective into an eigen-decomposition problem via theoretical analysis, and develop efficiency techniques in GEBE to produce BNE results without materializing all MHS and MHP scores.
- We then develop GEBE^p to solve our Poisson-instantiated BNE objective by leveraging the properties of Poisson distributions and randomized singular value decomposition, which further boosts efficiency and improves embedding quality.
- The superior performance of our proposed solutions is validated by comparing against 15 competitors on 10 real datasets.

Table 1: Frequently used notations.

Notation	Description
G=(U,V,E)	A bipartite graph G with node sets U, V and edge set E.
$\|U\|$, $\|V\|$	The number of nodes in U and V, respectively.
$\mathbf{W}$	The edge weight matrix of G.
$s(u_i, u_l)$	The MHS between nodes u_i and u_l in U (see Eq. (4)).
$\mathbf{P}$	The MHP matrix (see Eq. (5)).
$\mathbf{U}, \mathbf{V}$	The embedding vectors of nodes in U and V, respectively.
$\mathcal{L}(\mathbf{U}, \mathbf{V})$	The BNE objective function in Eq. (9).
ω	A probability mass function.
τ	The maximum length of paths.
k	The embedding dimensionality.

2 PROBLEM FORMULATION

2.1 Preliminaries

Let $G = (U, V, E)$ be a bipartite graph, where node sets U and V denote two types of nodes, and edge set $E \subset U \times V$ includes the inter-set edges between nodes in U and V. In a bipartite graph, edges only exist between nodes of different types, and each edge (u_i, v_j) is associated with a non-negative weight $w(u_i, v_j)$. Following convention [7, 24], we focus on undirected bipartite graphs.

Bipartite network embedding (BNE) maps every node $u_i \in U$ and every node $v_j \in V$, to low-dimensional length-k embedding vectors $\mathbf{U}[u_i]$ and $\mathbf{V}[v_j]$ respectively, such that $\mathbf{U}[u_i]$ and $\mathbf{V}[v_j]$ capture the hidden topological features surrounding nodes u_i and v_j respectively. We denote matrices $\mathbf{U} \in \mathbb{R}^{|U| \times k}$ and $\mathbf{V} \in \mathbb{R}^{|V| \times k}$ as the embedding vectors of all nodes in U and V, respectively. The embedding vectors are expected to achieve high accuracy in downstream tasks, such as top-N recommendation and link prediction.

Notations. We denote matrices in bold uppercase, *e.g.*, $\mathbf{U}$. We use $\mathbf{U}[i]$ or $\mathbf{U}[i, \cdot]$ to denote the i-th row vector of $\mathbf{U}$, and $\mathbf{U}[\cdot, j]$ to denote the j-th column vector of $\mathbf{U}$. We denote $\mathbf{U}[i, j]$ as the element at the i-th row and j-th column of $\mathbf{U}$. We use a weight matrix $\mathbf{W} \in \mathbb{R}^{|U| \times |V|}$ to represent all edge weights in G. If $(u_i, v_j) \in E$, $\mathbf{W}[u_i, v_j] = w(u_i, v_j)$; otherwise, $\mathbf{W}[u_i, v_j] = 0$. $\|\mathbf{U}\|_2$ and $\|\mathbf{U}\|_F$ represent the spectral norm and Frobenius norm of $\mathbf{U}$, respectively. Table 1 lists the frequently used notations in our paper.

2.2 Multi-Hop Homogeneous Similarity

As explained in Section 1, effective BNE should capture both the similarity between homogeneous nodes and the proximity between heterogeneous nodes. In this section, we first present a generic multi-hop homogeneous similarity (MHS) formula to evaluate the similarity of homogeneous nodes. For brevity, we use nodes u_i and u_l in U as an example, and the case of nodes in V is similar.

Given two nodes u_i and u_l in U, the length of any path connecting u_i and u_l must be even. In real-world bipartite graphs, there can be numerous paths connecting two nodes. Intuitively, we need to consider all multi-hop paths between u_i and u_l to evaluate the similarity between u_i and u_l.

Let $\xi = (u_i, v_{i_1}, u_{i_2}, \cdots, u_{i_{2\ell-2}}, v_{j_{2\ell-1}}, u_l)$ be a length-2ℓ path between u_i and u_l, where $\ell \geq 0$. Note that nodes in U and V appear alternatively in ξ since only inter-set edges exist in a bipartite

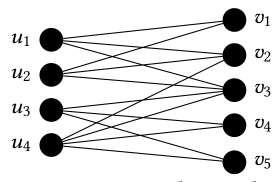

Figure 1: Example graph G

Table 2: H values of u_1, u_2, u_4.

	u_1	u_2	u_4
$\mathbf{H}[u_1,\cdot]$	3.641	3.506	4.064
$\mathbf{H}[u_2,\cdot]$	**3.506**	3.641	**4.064**
$\mathbf{H}[u_4,\cdot]$	4.064	4.064	5.429

graph. We define the *path weight* $w(\xi)$ of ξ as the product of all edge weights in the path, *i.e.*, $w(\xi) = w(u_i, v_{i_1}) \times w(v_{i_1}, u_{i_2}) \times \cdots \times w(v_{j_{2\ell-1}}, u_l)$. Given a specific ℓ, let $\Xi_{2\ell}$ be the set of all length-2ℓ paths between u_i and u_l. In Eq. (1), we sum up all length-2ℓ path weights and denote the result as $q_{2\ell}(u_i, u_l)$.

$$q_{2\ell}(u_i, u_l) = \sum_{\xi \in \Xi_{2\ell}} w(\xi), \tag{1}$$

where $\Xi_{2\ell}$ is the set of all length-2ℓ paths between u_i and u_l.

Then we rewrite Eq. (1) to its equivalent matrix form in Eq. (2).

$$q_{2\ell}(u_i, u_l) = (\mathbf{W}\mathbf{W}^\top)^\ell [u_i, u_l], \tag{2}$$

where $\mathbf{W}$ is the edge weight matrix of the input graph G.

Intuitively, to evaluate the similarity/proximity between nodes, we need to aggregate all paths between nodes, with different importance assigned based on path lengths. Therefore, we adopt a probability mass function (PMF), $\omega(\ell)$, to assign the path importance based on ℓ. In literature, there are many choices of $\omega(\ell)$, which is explained shortly in Section 2.4. Here we focus on the generic formula. Given a PMF $\omega(\ell)$, we perform weighted sum over all possible $q_{2\ell}(u_i, u_l)$ for $0 \le \ell \le \tau$, to get $\mathbf{H}(u_i, u_l)$, as shown in Eq. (3), where τ is the maximum ℓ considered.

$$\mathbf{H}[u_i, u_l] = \sum_{\ell=0}^{\tau} \omega(\ell) \cdot (\mathbf{W}\mathbf{W}^\top)^\ell [u_i, u_l]. \tag{3}$$

We could use $\mathbf{H}[u_i, u_l]$ to measure the similarity between u_i and u_l; however, we identify that $\mathbf{H}[u_i, u_l]$ suffers from an issue to measure the MHS of two homogeneous nodes, which may yield counter-intuitive similarity scores. In the following, we provide a running example to explain the issue and then propose a normalization technique to address it.

Running Example. Figure 1 shows a bipartite graph G with node sets $U = \{u_1, u_2, u_3, u_4\}$ and $V = \{v_1, v_2, v_3, v_4, v_5\}$. Assume that each edge has a weight 0.5. Table 2 lists the $\mathbf{H}[u_i, u_l]$ values of nodes u_1, u_2, u_4 computed when $\omega(\ell)$ is a Poisson distribution with a parameter $\lambda = 2$, in Eq. (8). Observe that in graph G, u_1 and u_2 share exactly the same neighbors $\{v_1, v_2, v_3\}$, while u_2 and u_4 have only two common neighbors $\{v_2, v_3\}$. For instance, if nodes in U are users, nodes in V are movies, and an edge represents a user's personal taste to a movie. Intuitively, user u_1 and u_2 have exactly the same taste, while u_2 and u_4 only share partial taste, meaning that the MHS between u_1 and u_2 should be larger than u_2 and u_4. However, the $\mathbf{H}$ values in Table 2 shows counter-intuitive results, where $\mathbf{H}[u_2, u_4] = 4.064$ is larger than $\mathbf{H}[u_2, u_1] = 3.506$. This issue indicates that we cannot simply use $\mathbf{H}$ as MHS scores, and it is caused by the fact that u_4 has more connections/paths in the graph than other nodes, making $\mathbf{H}[u_4,\cdot]$ tend to be larger.

Therefore, directly using $\mathbf{H}[u_i, u_l]$ as the MHS measure may introduce unwanted distortion when evaluating similarity, especially in real-world bipartite graphs, where the node degree distribution is skewed [3]. To this end, instead of simply using $\mathbf{H}$ as MHS, we formally define the MHS $s(u_i, u_l)$ between nodes u_i and u_l in Eq. (4), in which, $s(u_i, u_l)$ is based on normalized $\mathbf{H}[u_i, u_l]$ over $\mathbf{H}[u_i, u_i]$ and $\mathbf{H}[u_l, u_l]$.

$$s(u_i, u_l) = \frac{\mathbf{H}[u_i, u_l]}{\sqrt{\mathbf{H}[u_i, u_i]} \cdot \sqrt{\mathbf{H}[u_l, u_l]}}. \tag{4}$$

MHS $s(u_i, u_l)$ eliminates the distortion introduced by those nodes with many connections in the input graph. Revisit the above example, we have $s(u_1, u_2) = 0.981 > s(u_2, u_4) = 0.914$, which agrees with the intuition that u_1 and u_2 are more similar since they share exactly the same neighbors. In the following Lemma 2.1[1], we prove the necessary properties of MHS $s(u_i, u_l)$.

Lemma 2.1. *Given a bipartite graph $G = (U, V, E)$ and two nodes $u_i, u_l \in U$, MHS $s(u_i, u_l)$ satisfies the following properties:*
(i) $0 \le s(u_i, u_l) \le 1$,
(ii) $\forall u_i \in U, s(u_i, u_i) = 1$,
(iii) $s(u_i, u_l) = 0$ if there are no paths from u_i to u_l.

2.3 Multi-Hop Heterogeneous Proximity

In this section, we present the generic multi-hop heterogeneous proximity (MHP) formula to evaluate the strength of connections between nodes of different types. The length of any path connecting nodes $u_i \in U$ and $v_j \in V$ is odd, due to the fact that edges only exist between heterogeneous nodes in a bipartite graph. Since we have introduced $\mathbf{H}$ in Eq. (3), the MHP $\mathbf{P}[u_i, v_j]$ between $u_i \in U$ and $v_j \in V$ is defined naturally as follows by extending $\mathbf{H}$ with $\mathbf{W}$ in Eq. (5).

$$\mathbf{P} = \mathbf{H}\mathbf{W} = \sum_{\ell=0}^{\tau} \omega(\ell) \cdot (\mathbf{W}\mathbf{W}^\top)^\ell \mathbf{W}. \tag{5}$$

Note that MHP $\mathbf{P}[u_i, v_j]$ describes the strength of multi-hop connections between heterogeneous nodes u_i and v_j, and it is different from the MHS measure introduced above.

2.4 Instantiation of MHS and MHP

Recall that both MHS and MHP in Eq. (4) and (5) rely on $\mathbf{H}$ defined in Eq. (3). When computing $\mathbf{H}$, there is a PMF $\omega(\ell)$ to decide the path importance based on ℓ. In literature, there are three popular choices of $\omega(\ell)$, including *Uniform distribution* applied in uniform high-order proximity [72], *Geometric distribution* used in PPR [34], and *Poisson distribution* used in HKPR [12]. Thus, we choose these three distributions to instantiate $\mathbf{H}$.

$$\mathbf{H}_\tau = \sum_{\ell=0}^{\tau} \frac{1}{\tau} \cdot (\mathbf{W}\mathbf{W}^\top)^\ell, \tag{6}$$

In particular, in Eq. (6), $\mathbf{H}_\tau$ with Uniform distribution assigns all paths with the same $\frac{1}{\tau}$ importance regardless of path lengths.

$$\mathbf{H}_\alpha = \sum_{\ell=0}^{\tau=\infty} \alpha(1-\alpha)^\ell \cdot (\mathbf{W}\mathbf{W}^\top)^\ell, \tag{7}$$

[1]Missing proofs appear in Appendix A.

In Eq. (7), $\mathbf{H}_\alpha$ with Geometric distribution has a decay factor $\alpha \in (0,1)$ to assign more importance to shorter paths.

$$\mathbf{H}_\lambda = \sum_{\ell=0}^{\tau=\infty} \frac{e^{-\lambda}\lambda^\ell}{\ell!} \cdot (\mathbf{W}\mathbf{W}^\top)^\ell. \tag{8}$$

And in Eq. (8), $\mathbf{H}_\lambda$ with Poisson distribution has a parameter λ (a positive integer) to control path importance.

2.5 A Unified Objective Function

In this section, we formulate our BNE objective as an optimization problem that unifies and preserves the MHP and MHS of all possible pairs in a bipartite graph. The objective function is generic to support all the three instantiations introduced ahead.

Given the embedding dimensionality k, our objective is to learn (i) an embedding vector $\mathbf{U}[u_i]$ for each node u_i in U, and an embedding vector $\mathbf{V}[v_j]$ for each node v_j in V, such that the following objective is minimized:

$$\min_{\mathbf{U}\in\mathbb{R}^{|U|\times k}, \mathbf{V}\in\mathbb{R}^{|V|\times k}} \mathcal{L}(\mathbf{U},\mathbf{V}),$$

$$\mathcal{L}(\mathbf{U},\mathbf{V}) = \frac{1}{|U|\cdot|V|} \sum_{u_i\in U, v_j\in V} \left(\mathbf{U}[u_i]\cdot\mathbf{V}[v_j] - \mathbf{P}[u_i,v_j]\right)^2 \tag{9}$$
$$+\frac{1}{|U|^2}\sum_{u_i,u_l\in U}\left(\left\|\frac{\mathbf{U}[u_i]}{\|\mathbf{U}[u_i]\|_2} - \frac{\mathbf{U}[u_l]}{\|\mathbf{U}[u_l]\|_2}\right\|_2^2 - 2(1-s(u_i,u_l))\right)^2$$

The rationale of the above objective function is as follows. The first part of the function aims to ensure that for any two heterogeneous nodes $u_i \in U$ and $v_j \in V$, their MHP $\mathbf{P}[u_i,v_j]$ is preserved by the dot product of their respective embedding vectors, *i.e.*, $\mathbf{U}[u_i]\cdot\mathbf{V}[v_j]$. This is because $\mathbf{U}[u_i]\cdot\mathbf{V}[v_j]$ is typically used in downstream applications (*e.g.*, recommendation services [7, 24, 65, 77]) to measure the strength of association between u_i and v_j. For example, if U is a set of users and V is a set of movies, then $\mathbf{U}[u_i]\cdot\mathbf{V}[v_j]$ could be used to decide whether we should recommend movie v_j to user u_i.

Meanwhile, the second part of our objective function is to ensure that for any two homogeneous node $u_i, u_l \in U$, if their MHS score $s(u_i,u_l)$ is large, then their respective normalized embedding vectors should be *similar*, in the sense that $\frac{\mathbf{U}[u_i]}{\|\mathbf{U}[u_i]\|_2}$ and $\frac{\mathbf{U}[u_l]}{\|\mathbf{U}[u_l]\|_2}$ should have a small L_2 distance. The reason is that downstream applications often use the normalized embedding vector of each node u as a feature vector of u for classification tasks [7, 24, 26, 50]. Therefore, if two nodes u_i and u_l have a high MHS score, we would like their normalized embedding vectors to be similar, so that the classification results derived from the vectors would also be similar.

One may question why our objective function does not consider the MHS $s(v_j,v_h)$ for homogeneous nodes $v_j,v_h \in V$. The answer is that, by minimizing Eq. (9), we can implicitly ensure that the embedding vectors of v_j and v_h tend to be similar when their MHS $s(v_j,v_h)$ is large, as shown in the following lemma.

Lemma 2.2. *Given embeddings* $\mathbf{U}$ *and* $\mathbf{V}$ *such that* $\mathcal{L}(\mathbf{U},\mathbf{V})=0$, *then for any two nodes* v_j *and* v_h *in* V, *their embedding vectors* $\mathbf{V}[v_i],\mathbf{V}[v_j]$ *satisfy* $\frac{1}{2}\left\|\frac{\mathbf{V}[v_j]}{\|\mathbf{V}[v_j]\|_2} - \frac{\mathbf{V}[v_h]}{\|\mathbf{V}[v_h]\|_2}\right\|_2^2 = 1 - s(v_j,v_h)$, *where* $s(v_j,v_h) = \frac{\sum_{\ell=1}^{\tau}\omega(\ell)\cdot(\mathbf{W}^\top\mathbf{W})^\ell[v_j,v_h]}{\sqrt{\sum_{\ell=1}^{\tau}\omega(\ell)\cdot(\mathbf{W}^\top\mathbf{W})^\ell[v_j,v_j]}\cdot\sqrt{\sum_{\ell=1}^{\tau}\omega(\ell)\cdot(\mathbf{W}^\top\mathbf{W})^\ell[v_h,v_h]}}$, *similar to Eq. (4).*

3 SOLUTION OVERVIEW

It is technically challenging to obtain the embedding results of nodes based on our objective function in Eq. (9), especially on massive bipartite graphs. First, it requires the MHP and MHS scores of all possible node pairs in the input bipartite graph G, which are expensive to obtain, since MHS and MHP are defined over numerous multi-hop paths between nodes. Second, materializing all MHP and MHS scores requires $O(|U|\cdot|V| + |U|^2)$ space cost, which is infeasible for a large graph. Third, our objective function preserves both the MHS scores between homogeneous nodes and the MHP scores between heterogeneous nodes, which may make the training process hard to converge by using conventional gradient descent [14].

Toward this end, we first conduct rigorous theoretical analysis here and convert the optimization problem in Eq. (9) to an eigendecomposition computation in a nontrivial way. To facilitate analysis, assume that we have matrix $\mathbf{H}$ and its full exact eigenvectors $\mathbf{Z}\in\mathbb{R}^{|U|\times|U|}$ and eigenvalues $\mathbf{\Lambda}\in\mathbb{R}^{|U|\times|U|}$ available. We define

$$\mathbf{X} = \mathbf{Z}\sqrt{\mathbf{\Lambda}} \in \mathbb{R}^{|U|\times|U|}, \text{ and } \mathbf{Y} = \mathbf{W}^\top\mathbf{X} \in \mathbb{R}^{|V|\times|U|}, \tag{10}$$

Obviously, we have $\mathbf{H} = \mathbf{X}\cdot\mathbf{X}^\top$ and $\mathbf{X}\cdot\mathbf{Y}^\top = \mathbf{H}\mathbf{W} = \mathbf{P}$. In other words, for nodes $u_i, u_l \in U$ and $v_j \in V$, we have

$$\mathbf{H}[u_i,u_l] = \mathbf{X}[u_i]\cdot\mathbf{X}[u_l],\ \mathbf{P}[u_i,v_j] = \mathbf{X}[u_i]\cdot\mathbf{Y}[v_j]. \tag{11}$$

Further, we obtain

$$\begin{aligned}1 - s(u_i,u_l) &= 1 - \frac{\mathbf{H}[u_i,u_l]}{\sqrt{\mathbf{H}[u_i,u_i]]}\cdot\sqrt{\mathbf{H}[u_l,u_l]}}\\ &= 1 - \frac{\mathbf{X}[u_i]\cdot\mathbf{X}[u_l]}{\sqrt{\mathbf{X}[u_i]\cdot\mathbf{X}[u_i]}\cdot\sqrt{\mathbf{X}[u_l]\cdot\mathbf{X}[u_l]}}\\ &= \frac{1}{2}\cdot\left(\left\|\frac{\mathbf{X}[u_i]}{\|\mathbf{X}[u_i]\|_2}\right\|_2^2 + \left\|\frac{\mathbf{X}[u_l]}{\|\mathbf{X}[u_l]\|_2}\right\|_2^2\right) - \frac{\mathbf{X}[u_i]\cdot\mathbf{X}[u_l]}{\|\mathbf{X}[u_i]\|_2\cdot\|\mathbf{X}[u_l]\|_2}\\ &= \frac{1}{2}\cdot\left\|\frac{\mathbf{X}[u_i]}{\|\mathbf{X}[u_i]\|_2} - \frac{\mathbf{X}[u_l]}{\|\mathbf{X}[u_l]\|_2}\right\|_2^2.\end{aligned} \tag{12}$$

Using Eq. (11) and (12), we can get $\mathcal{L}(\mathbf{X},\mathbf{Y}) = 0$ in Eq. (9), meaning that the objective is optimized using Eq. (10) as a solution.

However, the full exact eigenvectors and eigenvalues of $\mathbf{H}$ are hard to obtain, and embedding vectors $\mathbf{U},\mathbf{V}$ in our objective are required to be k-dimensional. Therefore, we resort to use the top-k largest eigenvalues and eigenvectors of $\mathbf{H}$ to approximately solve our BNE objective. Specifically, according to Eq. (3), $\mathbf{H}$ is a non-negative symmetric matrix, implying that its eigenvalues $\mathbf{\Lambda}$ and all entries in eigenvectors $\mathbf{Z}$ are real numbers [25]. Then we prove Theorem 3.1, indicating that once we have the top-k eigenvalues $\mathbf{\Lambda}_k$ and respective eigenvectors $\mathbf{Z}_k$ of $\mathbf{H}$, we can derive an approximate solution to our objective in Eq. (9).

Theorem 3.1. *Given a bipartite graph* $G = (U,V,E)$, *we define* $\mathbf{U}^*\in\mathbb{R}^{|U|\times k}, \mathbf{V}^*\in\mathbb{R}^{|V|\times k}$ *as follows:*

$$\mathbf{U}^* = \mathbf{Z}_k\sqrt{\mathbf{\Lambda}_k} \text{ and } \mathbf{V}^* = \mathbf{W}^\top\mathbf{U}^*, \tag{13}$$

where $\mathbf{Z}_k$ *contains the exact top-k eigenvectors of* $\mathbf{H}$ *and* $\mathbf{\Lambda}_k$ *is a diagonal matrix where each diagonal entry is an exact top-k eigenvalue of* $\mathbf{H}$. *Then* $\mathbf{U}^*,\mathbf{V}^*$ *is an approximate solution to our objective in Eq. (9) and the loss is bounded by*

$$\mathcal{L}(\mathbf{U}^*,\mathbf{V}^*) \le \frac{\sigma_{k+1}^2}{|U|}\left(\frac{\sum_{(u_i,v_j)\in E} w(u_i,v_j)^2}{|V|} + \frac{\sum_{u_i\in U}\frac{2}{(\mathbf{H}[u_i,u_i]-\sigma_{k+1})^2}}{|U|}\right),$$

where σ_{k+1} denotes the $(k+1)$-th largest singular value of $\mathbf{H}$.

However, in practice, it is still challenging to leverage Theorem 3.1 to approximate our BNE objective since Theorem 3.1 still requires matrix $\mathbf{H}$, while the computation of $\mathbf{H}$ involves numerous iterations, each of which needs $O(|E|\cdot|U|)$ time. Further, it requires $O(|U|^2)$ space to store $\mathbf{H}$. Moreover, the top-k eigenvalues and eigenvectors of $\mathbf{H}$ are hard to obtain, and a typical solver needs $O(|U|^2k)$ time per iteration, which is prohibitive for massive graphs.

To alleviate these challenges, in Section 4, we first present GEBE, an efficient solution that optimizes our BNE objective and outputs high-quality embeddings. In particular, GEBE does not need to materialize $\mathbf{H}$, and employs *Krylov subspace iterations* and *power iteration* to approximate eigenvectors. Moreover, GEBE is general to support all the three instantiations introduced in Section 2.4. In experiments, we observe that Poisson-instantiated GEBE almost always produces the best result quality over all datasets, and thus, in Section 5, we further present a more efficient algorithm, GEBE^p, which is specialized for Poisson distribution with a rigorous approximation guarantee.

4 A GENERAL SOLUTION: GEBE

As explained in Section 3, to approximate our BNE objective, it requires the top-k eigenvalues and eigenvectors of matrix $\mathbf{H}$ in Eq. (3). A naive solution is to materialize $\mathbf{H}$, and then compute the eigenvalues and eigenvectors. However, this is inefficient in terms of both running time and space cost. For the purpose of efficiency, in this section, we present the technical details of GEBE that combines *Krylov subspace iterations* (KSI) [25] and *power iteration* method [49] to generate BNE results without actually materializing $\mathbf{H}$.

4.1 Algorithm

We first review the idea of KSI. Given matrix $\mathbf{H}$, and starting with a randomly initialized semi-unitary matrix $\mathbf{Z}'_k \in \mathbb{R}^{|U|\times k}$, KSI iteratively computes $\mathbf{HZ}'_k$ and applies QR decomposition on $\mathbf{HZ}'_k$ to get a new $\mathbf{Z}'_k$ for next iteration; this process repeats until $\mathbf{Z}'_k$ converges or a predefined number of iterations is reached, *e.g.*, $t = 200$. Then $\mathbf{Z}'_k$ is returned as the *approximate* top-k eigenvectors.

However, recall that $\mathbf{H}$ is expensive to materialize. We observe that in the KSI computation explained above, it actually uses $\mathbf{HZ}'_k \in \mathbb{R}^{|U|\times k}$ instead of $\mathbf{H} \in \mathbb{R}^{|U|\times|U|}$, where $k \ll |U|$ in practice. Therefore, we propose to directly materialize $\mathbf{HZ}'_k$ instead of the expensive $\mathbf{H}$, based on the following equation, which can be efficiently implemented by power iteration method.

$$\mathbf{HZ}'_k = \sum_{\ell=0}^{\tau} \omega(\ell)\cdot(\mathbf{WW}^\top)^\ell \mathbf{Z}'_k \tag{14}$$

Algorithm 1 shows the pseudo-code of GEBE that takes as input a bipartite graph G, a PMF $\omega(\ell)$, embedding size k, the number of iterations t used in KSI, and τ the maximum ℓ considered. Starting with a randomly picked semi-unitary matrix $\mathbf{Z}'_k$ at Line 1, from Line 2 to Line 7, within t iterations or until convergence, GEBE first computes $\mathbf{HZ}'_k$ with the help of an intermediate matrix $\mathbf{Q}$ by power

Algorithm 1: GEBE

```
Input: bipartite graph $G$, PMF $\omega$, embedding size $k$, number of
       iterations $t, \tau$.
Output: embedding vectors $\mathbf{U}, \mathbf{V}$.
1  Randomly pick a matrix $\mathbf{Z}'_k \in \mathbb{R}^{|U|\times k}$ such that $\mathbf{Z}'^\top_k \mathbf{Z}'_k = \mathbf{I} \in \mathbb{R}^{k\times k}$;
2  for $i \leftarrow 1$ to $t$ do
3      $\mathbf{Q}_0 \leftarrow \mathbf{Z}'_k$; $\mathbf{Q} \leftarrow \omega(0)\cdot\mathbf{Q}_0$;
4      for $\ell \leftarrow 1$ to $\tau$ do
5          $\mathbf{Q}_\ell \leftarrow \mathbf{W}\cdot(\mathbf{W}^\top\mathbf{Q}_{\ell-1})$;
6          $\mathbf{Q} \leftarrow \mathbf{Q} + \omega(\ell)\cdot\mathbf{Q}_\ell$;
7      Apply QR decomposition over $\mathbf{Q}$ to obtain the updated $\mathbf{Z}'_k$ and
         an upper-triangular matrix $\mathbf{R}$;
8  Initialize an empty $k\times k$ diagonal matrix $\mathbf{\Lambda}'_k$;
9  for $i \leftarrow 1$ to $k$ do
10     $\mathbf{\Lambda}'_k[i,i] \leftarrow \mathbf{R}[i,i]$;
11 $\mathbf{U} \leftarrow \mathbf{Z}'_k\cdot\sqrt{\mathbf{\Lambda}'_k}$; $\mathbf{V} \leftarrow \mathbf{W}^\top\cdot\mathbf{U}$;
12 return $\mathbf{U}, \mathbf{V}$;
```

iteration (Lines 3 - 6), and then updates approximate top-k eigenvectors $\mathbf{Z}'_k$ and gets a byproduct matrix $\mathbf{R}$ by QR decomposition at Line 7. Note that the computation from Lines 3 to 6 is equivalent to Eq. (14), with an efficiency trick to reorder the matrix multiplication $\mathbf{WW}^\top\mathbf{Q}_{\ell-1}$ to $\mathbf{W}\cdot(\mathbf{W}^\top\mathbf{Q}_{\ell-1})$, thereby reducing $O(|E|\cdot|U|)$ time cost to $O(|E|\cdot k)$. After getting the top-k eigenvectors $\mathbf{Z}'_k$, note that the first k diagonal elements of the byproduct matrix $\mathbf{R}$ are actually the corresponding eigenvalues [17], and we extract such eigenvalues into a $k\times k$ diagonal matrix $\mathbf{\Lambda}'_k$ at Lines 8-10. Then at Lines 11 and 12, based on Eq. (13), we calculate $\mathbf{U} = \mathbf{Z}'_k\sqrt{\mathbf{\Lambda}'_k}$, $\mathbf{V} = \mathbf{W}^\top\mathbf{U}$, and return them as the embedding results.

4.2 Analysis

Theorem 4.1 proves the correctness of Algorithm 1.

Theorem 4.1. *Let* $\mathbf{U}$ *and* $\mathbf{V}$ *be the outputs of Algorithm 1 when* $\mathbf{Z}'_k$ *converges within t KSI iterations in Algorithm 1. Then,* $\mathbf{U} = \mathbf{U}^*$ *and* $\mathbf{V} = \mathbf{V}^*$*, where* $\mathbf{U}^*, \mathbf{V}^*$ *are defined in Eq.* (13).

GEBE needs to materialize matrices $\mathbf{W}, \mathbf{Z}'_k, \mathbf{Q}, \mathbf{U}$ and $\mathbf{V}$, with space cost bounded by $O((|U|+|V|)\cdot k+|E|)$. The computation of $\mathbf{Q}$ at Lines 3-6 in Algorithm 1 requires $O(k\tau|E|)$ time. The QR decomposition over $\mathbf{Q} \in \mathbb{R}^{|U|\times k}$ at Line 7 takes $O(|U|\cdot k^2)$ time. Thus, the overall t KSI iterations require $O(kt\tau|E| + k^2t|U|)$ time. The computation of $\mathbf{U}$ and $\mathbf{V}$ at Line 11 needs $O(|U|\cdot k^2)$ time and $O(|E|\cdot k)$ time, respectively. Therefore, the time complexity of GEBE (Algorithm 1) is $O(kt\tau|E| + k^2t|U|)$.

Connection to NRP. In what follows, we analyze the connection between NRP and GEBE. In particular, given a bipartite graph G with node sets U and V, NRP constructs two embedding matrices $\mathbf{F} \in \mathbb{R}^{(|U|+|V|)\times\frac{k}{2}}$ and $\mathbf{B} \in \mathbb{R}^{\frac{k}{2}\times(|U|+|V|)}$, such that for any two nodes x, y, we have

$$\mathbf{F}[x,\cdot]\cdot\mathbf{B}[\cdot,y] \approx \overrightarrow{\omega}_x\cdot\overleftarrow{\omega}_y\cdot\sum_{\ell=1}^{\infty}\alpha(1-\alpha)^\ell\cdot\mathbf{T}^\ell[x,y],$$

where $\mathbf{T} = \left[\begin{array}{c|c} \mathbf{0} & \mathbf{W}_r \\ \hline \mathbf{W}_c^\top & \mathbf{0} \end{array}\right] \in \mathbb{R}^{(|U|+|V|)\times(|U|+|V|)}$, while $\mathbf{W}_r$ (resp. $\mathbf{W}_c$) is the row-normalized (resp. column-normalized) version of the weight matrix $\mathbf{W}$, and $\overrightarrow{\omega}_x$ (resp. $\overleftarrow{\omega}_y$) is a positive constant associated with x (resp. y). In other words, the embedding matrices $\mathbf{F}$ and $\mathbf{B}$ produced by NRP form a low-rank approximation of a personalized PageRank (PPR) matrix weighted by $\overrightarrow{\omega}_x$ and $\overleftarrow{\omega}_y$. Intuitively, this is similar in spirit to how GEBE preserves the multi-hop heterogeneous proximity (MHP) $\mathbf{P}[u,v]$ between two nodes $u \in U$ and $v \in V$: as shown in Section 2.5, GEBE aims to ensure that

$$\mathbf{U}[u] \cdot \mathbf{V}[v] \approx \mathbf{P}[u,v],$$

where $\mathbf{U}[u]$ and $\mathbf{V}[v]$ are the embedding vector of nodes u and v, respectively. The crucial difference, however, is that GEBE also aims to preserve the multi-hop homogeneous similarity (MHS) $s(u_i,u_l)$ between any two different nodes $u_i, u_l \in U$, by requiring that

$$\left\| \frac{\mathbf{U}[u_i]}{\|\mathbf{U}[u_i]\|_2} - \frac{\mathbf{U}[u_l]}{\|\mathbf{U}[u_l]\|_2} \right\|_2^2 \approx 2(1 - s(u_i,u_l)). \quad (15)$$

In other words, it aims to ensure that if the MHS $s(u_i,u_l)$ between u_i and u_l is large, then their normalized embedding vectors should be similar. (By Lemma 2.2, our solution ensures that Eq. (15) also holds for any two nodes $v_i, v_l \in V$.) This helps improve the accuracy of classification in the bipartite graph, which is explained in Section 2.5, and is empirically demonstrated in Table 5 in Section 6.4. In summary, GEBE preserves MHP in a way somewhat similar to how NRP approximates weighted personalized PageRanks, but the former achieves much better effectiveness by taking into account the MHS between nodes, which is not considered in NRP.

5 A SPECIALIZED SOLUTION: GEBE^p

In Section 6, we evaluate the performance of GEBE instantiated with $\mathbf{H}_\tau$, $\mathbf{H}_\alpha$, and $\mathbf{H}_\lambda$ introduced in Section 2.4, and find that GEBE with the Poisson-based $\mathbf{H}_\lambda$ yields the best accuracy in most cases. Therefore, in this section, by leveraging the properties of Poisson distribution, we further propose GEBE^p, a specialized solution with a novel and efficient design, to solve our Poisson-instantiated BNE objective. GEBE^p is demonstrated to be much more efficient and effective than GEBE in our experiments.

5.1 Algorithm

Note that when instantiated with Poisson distribution, GEBE in Algorithm 1 requires a parameter τ as defined in Eq. (8), where theoretically τ should be ∞. However, in practice, we set $\tau = 20$ when running GEBE, to trade effectiveness for efficiency. Moreover, the KSI computation at Line 2 of Algorithm 1 requires a large number of iterations to converge. GEBE^p addresses these deficiencies of GEBE, and significantly improves the BNE performance. Moreover, GEBE^p offers a rigorous approximation guarantee of the returned embedding vectors $\mathbf{U}$ and $\mathbf{V}$ to the optimal solution $\mathbf{U}^*$ and $\mathbf{V}^*$ of the optimization problem in Eq. (9).

We first explain the idea behind GEBE^p, and then present the detailed algorithm. For any real square matrix $\mathbf{M}$, we can get $e^{\mathbf{M}} = \sum_{k=0}^{\infty} \frac{\mathbf{M}^k}{k!}$ [46]. Let $\mathbf{M} = \lambda\mathbf{W}\mathbf{W}^\top$. Then, we have

$$\frac{e^{\lambda\mathbf{W}\mathbf{W}^\top}}{e^\lambda} = \sum_{\ell=0}^{\infty} \frac{e^{-\lambda}\lambda^\ell}{\ell!}(\mathbf{W}\mathbf{W}^\top)^\ell = \mathbf{H}_\lambda. \quad (16)$$

Algorithm 2: GEBE^p

```
Input: bipartite graph $G$, embedding size $k$, parameter $\lambda$, error
       threshold $\epsilon$.
Output: embedding vectors $\mathbf{U}, \mathbf{V}$.
1 Invoke RandomizedSVD($\mathbf{W}, k, \epsilon$) to obtain approximate left
    singular vectors $\mathbf{\Phi}'_k$ and its corresponding eigenvalues $\Sigma'_k$;
2 $\Lambda'_k \leftarrow e^{-\lambda} \cdot e^{\lambda\Sigma'^2_k}$;
3 $\mathbf{Z}'_k \leftarrow \mathbf{\Phi}'_k$;
4 $\mathbf{U} \leftarrow \mathbf{Z}'_k\sqrt{\Lambda'_k}$; $\mathbf{V} \leftarrow \mathbf{W}^\top\mathbf{U}$;
5 return $\mathbf{U}, \mathbf{V}$;
```

Let $\mathbf{\Phi} \in \mathbb{R}^{|U|\times|V|}$, $\Sigma \in \mathbb{R}^{|V|\times|V|}$ and $\mathbf{\Psi} \in \mathbb{R}^{|V|\times|V|}$ be the full left singular vectors, singular values and right singular vectors of matrix $\mathbf{W}$, respectively. Also let $\Sigma[i,i]$ and $\mathbf{\Phi}[\cdot,i]$ signify the i-th largest eigenvalue and eigenvector of $\mathbf{W}$, respectively. Note that $\mathbf{\Psi}$ is a unitary matrix, *i.e.*, $\mathbf{\Psi}^\top\mathbf{\Psi} = \mathbf{I}$ [25]. Then, based on the property of matrix exponential, we obtain

$$e^\lambda\mathbf{H}_\lambda = e^{\lambda\mathbf{W}\mathbf{W}^\top} = e^{\lambda(\mathbf{\Phi}\Sigma\mathbf{\Psi}^\top)\cdot(\mathbf{\Phi}\Sigma\mathbf{\Psi}^\top)^\top} = e^{\lambda\mathbf{\Phi}\Sigma^2\mathbf{\Phi}^\top} = \mathbf{\Phi}e^{\lambda\Sigma^2}\mathbf{\Phi}^\top.$$

Note that $\mathbf{\Phi}$ is also a unitary matrix, *i.e.*, $\mathbf{\Phi}^\top\mathbf{\Phi} = \mathbf{I}$. We further have, for every $1 \le i \le |V|$,

$$\mathbf{H}_\lambda \cdot \mathbf{\Phi}[\cdot,i] = e^{-\lambda} \cdot \mathbf{\Phi}e^{\lambda\Sigma^2}\mathbf{\Phi}^\top \cdot \mathbf{\Phi}[\cdot,i] = e^{-\lambda} \cdot e^{\lambda\Sigma[i,i]^2} \cdot \mathbf{\Phi}[\cdot,i], \quad (17)$$

which indicates that $e^{-\lambda} \cdot e^{\lambda\Sigma[i,i]^2}$ is an eigenvalue of $\mathbf{H}_\lambda$ and $\mathbf{\Phi}[\cdot,i]$ is its corresponding eigenvector.

Recall that $\Sigma[i,i]$ represents the i-th largest eigenvalue of $\mathbf{W}$ and singular values are non-negative. Thus, we conclude that $e^{-\lambda} \cdot e^{\lambda\Sigma[i,i]^2}$ is the i-th largest eigenvalue of $\mathbf{H}_\lambda$, and $\mathbf{\Phi}[\cdot,i]$ is also the i-th largest eigenvector of $\mathbf{H}_\lambda$. In other words, to get the top-k eigenvalues and eigenvectors of $\mathbf{H}_\lambda$, we only need to obtain matrices $\mathbf{\Phi}$, Σ of $\mathbf{W}$. Therefore, with all above careful analysis, the computation of $\mathbf{H}_\lambda$'s top-k eigenvalues and eigenvectors can be converted to perform singular value decomposition (SVD) over edge weight matrix $\mathbf{W}$. Note that SVD over $\mathbf{W}$ is efficient since $\mathbf{W}$ is sparse with $O(|E|)$ nonzero entries.

Algorithm 2 illustrates the pseudo-code of GEBE^p, which takes as input G, λ, k, and an error parameter ϵ used in SVD. Compared with GEBE in Algorithm 1, a key difference is that GEBE^p does not need parameters τ and t. At Line 1, we employ RandomizedSVD [47] to factorize edge weight matrix $\mathbf{W}$ of G with dimension k and error threshold ϵ, in order to get approximated left singular matrix $\mathbf{\Phi}'_k \in \mathbb{R}^{|U|\times k}$ and the corresponding eigenvalues in diagonal matrix $\Sigma'_k \in \mathbb{R}^{k\times k}$. Then we set $\mathbf{Z}'_k = \mathbf{\Phi}'_k$ and $\Lambda'_k = e^{-\lambda} \cdot e^{\lambda\Sigma'^2}$ at Lines 2-3. Finally, GEBE^p computes embeddings $\mathbf{U} = \mathbf{Z}'_k\Lambda'_k$ and $\mathbf{V} = \mathbf{W}^\top\mathbf{U}$ at Line 4, according to Eq. (13), and returns results at Line 5.

5.2 Analysis

Theorem 5.1 establishes the approximation guarantee of the embedding results $\mathbf{U}, \mathbf{V}$ returned by GEBE^p.

THEOREM 5.1. *Given a bipartite graph $G = (U, V, E)$, embedding dimensionality $k \in (0, \min\{|U|, |V|\})$ and an error threshold ϵ for*

Table 3: Statistics of Datasets.

Name	$\lvert U \rvert$	$\lvert V \rvert$	$\lvert E \rvert$	Type
DBLP	6,001	1,308	29,256	weighted
Wikipedia	15,000	3,214	64,095	unweighted
Pinterest	55,187	9,916	1,500,809	unweighted
Yelp	31,668	38,048	1,561,406	unweighted
MovieLens	69,878	10,677	10,000,054	weighted
Last.fm	359,349	160,168	17,559,530	weighted
MIND	876,956	97,509	18,149,915	unweighted
Netflix	480,189	17,770	100,480,507	weighted
Orkut	2,783,196	8,730,857	327,037,487	unweighted
MAG	10,541,560	2,784,240	1,095,315,106	weighted

SVD, the embeddings $\mathbf{U}, \mathbf{V}$ *returned by Algorithm 2 satisfy that*

$$\|\mathbf{U}_\lambda^* \mathbf{U}_\lambda^* - \mathbf{U}\mathbf{U}\|_F^2 \le \sum_{i=1}^{k} \left(\frac{e^{\lambda \sigma_i^2}}{e^\lambda} - \frac{e^{\lambda(\sigma_i^2 - \epsilon \sigma_{k+1}^2)}}{e^\lambda} \right),$$

$$\|\mathbf{U}_\lambda^* \mathbf{V}_\lambda^* - \mathbf{U}\mathbf{V}\|_F^2 \le \sigma_1^2 \cdot \sum_{i=1}^{k} \left(\frac{e^{\lambda \sigma_i^2}}{e^\lambda} - \frac{e^{\lambda\left(\sigma_i^2 - \epsilon \sigma_{k+1}^2\right)}}{e^\lambda} \right),$$

where σ_i *is the i-th largest singular value of* $\mathbf{W}$*, and* $\mathbf{U}_\lambda^*, \mathbf{V}_\lambda^*$ *are the Poisson-instantiated version of* $\mathbf{U}^*, \mathbf{V}^*$ *defined in Eq.*(13) *with* $\mathbf{H} = \mathbf{H}_\lambda$.

Theorem 5.1 implies that GEBE^p provides better accuracy than GEBE (Poisson) does, because it achieves a better approximate solution of the objective function in Eq. (9). To explain, as aforementioned, GEBE (Poisson) generates embeddings by approximating the objective function in Eq. (9) based on the truncated versions of MHP/MHS with $\tau \le 20$. In contrast, GEBE^p directly derives embeddings based on the exact matrix $\mathbf{H}_\lambda$ defined in Eq. (8) by leveraging the property of the Poisson distribution. Consequently, GEBE^p is able to derive a more accurate solution to Eq. (9), leading to more effective embeddings, as shown in our experiments in Section 6.

The space overhead is determined by the sizes of $\mathbf{Z}_k', \mathbf{\Lambda}_k', \mathbf{U}, \mathbf{V}$ and $\mathbf{W}$. Hence, the space complexity is bounded by $O((|U|+|V|)\cdot k+|E|)$. As stated in [47], RandomizedSVD (Line 1 in Algorithm 2) runs in $O\left((|E| \cdot k + |U| \cdot k^2) \cdot \frac{\log(|V|)}{\epsilon}\right)$ time. The matrix multiplications at Lines 2-4 of Algorithm 2 require $O(|U| \cdot k^2 + |E| \cdot k)$ time in total. Therefore, the overall time complexity of GEBE^p in Algorithm 2 is $O\left((|E| \cdot k + |U| \cdot k^2) \cdot \frac{\log(|V|)}{\epsilon}\right)$.

6 EXPERIMENTS

We experimentally evaluate GEBE (with three instantiations) and GEBE^p against 15 competitors on top-N recommendation and link prediction tasks, using 10 real-world bipartite graphs. All experiments are conducted on a Linux machine powered by an Intel Xeon(R) Gold 6240@2.60GHz CPU and 377GB RAM. Source codes of all competitors are obtained from the respective authors, which are implemented in Python.

6.1 Experimental Setup

Datasets. Table 3 lists the statistics of the datasets used in our experiments. The datasets cover a full spectrum of various bipartite graphs, including social networks, academic graphs, web graphs, rating graphs, etc. All these bipartite graphs are used in prior work [6, 7, 9, 24, 30, 42, 66, 67]. DBLP[2] is a publication network with a set of authors, a set of venues, and a set of inter-set edges with weights indicating the number of papers that an author published to a venue. Wikipedia[2] contains authors and the pages that these authors edited as a bipartite graph. Each edge in Pinterest[3] denotes that a user has pinned an image to his/her own board. Yelp[4] builds the interactions between users and locations (*e.g.*, restaurants) as a bipartite graph. MovieLens[5] is a weighted user-movie rating bipartite graph. Last.fm[6] contains the play count of each music by each user. MIND[7] contains users' click history on news articles as a bipartite graph. Netflix[8] is also a weighted user-movie rating bipartite graph. Orkut[9] is a user-to-group affiliation bipartite network. MAG[10] is extracted based on the Microsoft Academic Graph [58], and contains a set of papers and a set of words extracted from abstracts as nodes, as well as a set of edges connecting papers and words with word occurrence as edge weights. There are 5 weighted bipartite graphs and 5 unweighted ones. Following convention [7, 24], we evaluate the top-N recommendation task on weighted bipartite graphs and evaluate the link prediction task on unweighted graphs.

Baselines and Parameter Settings. We compare GEBE and GEBE^p against 15 existing methods, including (i) BNE solutions: BiNE [24] and BiGI [7], (ii) homogeneous network embedding methods by regarding the input bipartite graph as a homogeneous graph: DeepWalk [50], node2vec [26], LINE [60], NRP [74], and (iii) collaborative filtering algorithms: BPR [53], NCF [31], NGCF [67], LightGCN [30], GCMC [5], CSE [9], LCFN [78], LR-GCCF [11], SCF [87]. To study the effects of MHP and MHS measures, we also add two baselines, MHP-BNE and MHS-BNE, which only optimize Poisson-instantiated MHP and MHS measures, respectively. Specifically, MHP-BNE preserves the MHP $\mathbf{P}[u_i, v_j]$ of all heterogeneous node pairs where $u_i \in U$, and $v_j \in V$. MHS-BNE preserves the MHS $s(u_i, u_l)$ of any two nodes u_i and u_l in U as well as the MHS $s(v_i, v_l)$ of any two nodes v_i and v_l in V.

All methods run on a single CPU core. Note that we evaluate three versions of GEBE based on the instantiations in Section 2.4, and denote them as GEBE (Uniform), GEBE (Geometric) and GEBE (Poisson), whose PMF $\omega(\ell)$ is a uniform PMF, a geometric PMF and a Poisson PMF, respectively. For all methods, we set the embedding dimensionality k as 128 for a fair comparison. Then the parameters of all competitors are set as suggested in their respective papers. Unless otherwise specified, we set $\alpha = 0.5$ for GEBE (Geometric), $\lambda = 1$ for MHP-BNE, MHS-BNE, GEBE (Poisson) and GEBE^p. Moreover, we set $t = 200$, $\tau = 20$ for MHP-BNE, MHS-BNE and all GEBE methods, and $\epsilon = 0.1$ for GEBE^p, respectively.

The efficiency evaluation of all methods and scalability evaluation of our proposed methods are reported in Section 6.2. We exclude a method if it cannot finish training within three days or it

[2]https://github.com/clhchtcjj/BiNE/tree/master/data/
[3]https://sites.google.com/site/xueatalphabeta/academic-projects
[4]https://github.com/kuandeng/LightGCN/tree/master/Data/yelp2018
[5]https://grouplens.org/datasets/movielens
[6]http://ocelma.net/MusicRecommendationDataset/lastfm-360K.html
[7]https://msnews.github.io
[8]https://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a
[9]http://networkrepository.com/aff-orkut-user2groups.php
[10]https://figshare.com/articles/dataset/mag_scholar/12696653

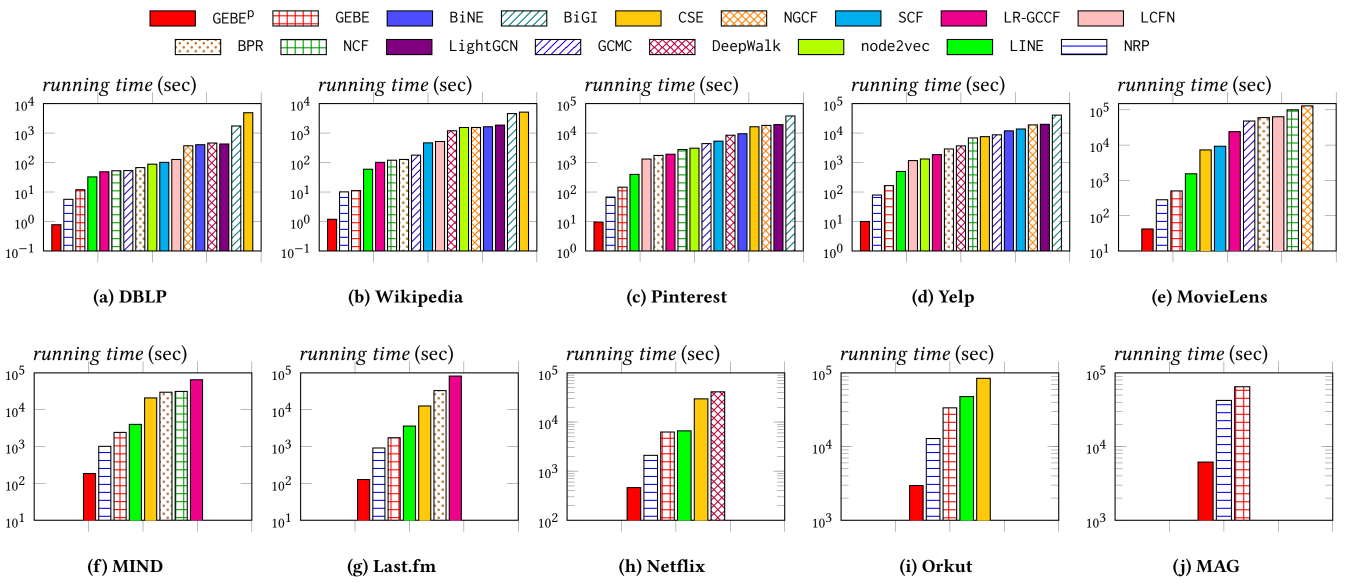

Figure 2: Running time (best viewed in color).

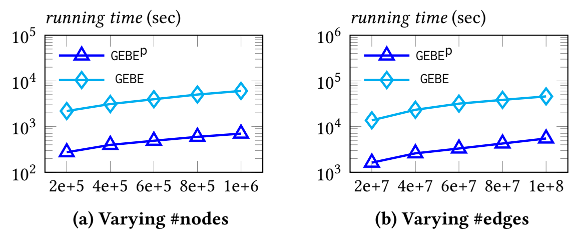

Figure 3: Scalability tests.

runs out of memory. The evaluation results of our proposed methods comparing against the competitors for top-N recommendation and link prediction are reported in Section 6.3 and Section 6.4, respectively. Further in Section 6.5, we report the evaluation results when varying the parameters of our method GEBE^p.

6.2 Efficiency and Scalability

Figure 2 reports the time required by all methods to construct embedding results on ten datasets. The y-axis is the running time (seconds) in log-scale. The reported running time excludes the time for loading datasets and outputting embedding vectors. We omit any methods with processing time exceeding three days or triggering out of memory errors.

As shown in Figure 2, GEBE^p, which solves our Poisson-based BNE objective in Algorithm 2, is consistently faster than all competitors on all datasets, often by orders of magnitudes. Specifically, our methods GEBE^p and GEBE are highly efficient on large bipartite graphs, *e.g.*, Orkut and MAG in Figures 2(i) and 2(j). In particular, on MAG with 13.3 million nodes and 1.1 billion edges, GEBE^p takes 1.7 hours to finish, while almost all competitors fail to return results within three days, and the fastest competitor NRP, costs 11.8 hours, meaning that GEBE^p is 6.9× faster. Note that NRP is designed for homogeneous graphs and neglects the important features of bipartite graphs. Therefore, compared with GEBE and GEBE^p, NRP achieves inferior accuracy in top-N recommendation and link prediction tasks, as demonstrated later in Sections 6.3 and 6.4. GEBE and GEBE^p are also efficient on small/moderate sized bipartite graphs in Figures 2(a)-2(h). For instance, on DBLP in Figure 2(a), GEBE takes 12 seconds to finish, and GEBE^p takes 0.8 seconds, which is 500× faster than BiNE (403 seconds), and 2174× faster than BiGI (1739 seconds), two representative BNE competitors. In summary, our proposed methods achieve significantly high efficiency for BNE computation over all types of bipartite graphs with various volumes. The high efficiency is achieved by the algorithmic designs proposed in Section 4 and 5.

We then report the scalability evaluation of GEBE and GEBE^p in Figure 3, when varying the number of nodes and edges in the input bipartite graph. Specifically, we apply the bipartite version of Erdős-Rényi random graph model [22] to generate synthetic bipartite graphs with various sizes. In Figure 3(a), we vary the number of nodes in $\{2\times10^5, 4\times10^5, 6\times10^5, 8\times10^5, 1\times10^6\}$ and fix the number of edges as 10^7, and report the respective running time of GEBE and GEBE^p. Observe that the runtime of the proposed methods increases almost linearly proportional to the number of nodes. Similarly, Figure 3(b) reports the runtime of GEBE and GEBE^p, when varying the number of edges in $\{2\times10^7, 4\times10^7, 6\times10^7, 8\times10^7, 1\times10^8\}$ while fixing the number of nodes as 10^6, from which, we observe that the runtime increases steadily with the number of nodes. The scalability tests demonstrate that our methods are scalable, and validate our time complexity analysis in Sections 4.2 and 5.2. We refer interested readers to Appendix B of our technical report [73] for the time complexity analysis of each method.

6.3 Top-N Recommendation

Given a user-item bipartite graph G, a user u, and an integer N, top-N recommendation task aims to recommend N items to user u, such that the N items are most likely to be attractive to user u. Top-N recommendation over weighted bipartite graphs is widely used

Table 4: Top-N ($N = 10$) recommendation performance

Method	DBLP			MovieLens			Last.fm			Netflix			MAG		
	F1	NDCG	MRR	F1	NDCG	MRR	F1	NDCG	MRR	F1	NDCG	MRR	F1	NDCG	MRR
GEBE^p	**0.214**	**0.261**	**0.601**	**0.266**	**0.304**	**0.558**	0.534	0.554	**0.778**	**0.342**	**0.366**	**0.611**	**0.265**	**0.286**	**0.468**
GEBE (Poisson)	0.212	0.258	0.581	0.256	0.294	0.548	0.53	0.549	0.766	0.334	0.354	0.586	0.261	0.284	0.465
GEBE (Geometric)	0.209	0.255	0.579	0.255	0.294	0.55	0.53	0.549	0.75	0.334	0.354	0.582	0.256	0.284	0.467
GEBE (Uniform)	0.209	0.255	0.579	0.253	0.292	0.55	0.53	0.55	0.755	0.334	0.353	0.58	0.255	0.282	0.464
MHP-BNE	0.203	0.249	0.547	0.225	0.265	0.52	0.501	0.528	0.743	-	-	-	-	-	-
MHS-BNE	0.126	0.141	0.303	0.176	0.208	0.48	0.357	0.4	0.676	-	-	-	-	-	-
BiNE	0.18	0.216	0.5	0.252	0.28	0.527	-	-	-	-	-	-	-	-	-
BiGI	0.191	0.23	0.526	0.255	0.293	0.536	-	-	-	-	-	-	-	-	-
DeepWalk	0.028	0.027	0.066	-	-	-	-	-	-	-	-	-	-	-	-
node2vec	0.032	0.029	0.063	-	-	-	-	-	-	-	-	-	-	-	-
LINE	0.061	0.063	0.16	0.242	0.262	0.477	0.182	0.196	0.315	0.155	0.166	0.284	-	-	-
NRP	0.191	0.244	0.567	0.212	0.236	0.464	0.509	0.542	0.745	0.287	0.308	0.534	0.222	0.23	0.436
BPR	0.209	0.253	0.568	0.258	0.278	0.519	0.497	0.511	0.709	-	-	-	-	-	-
NCF	0.185	0.209	0.493	0.212	0.215	0.458	-	-	-	-	-	-	-	-	-
NGCF	0.189	0.215	0.495	0.219	0.22	0.462	-	-	-	-	-	-	-	-	-
LightGCN	0.192	0.224	0.492	0.19	0.199	0.414	-	-	-	-	-	-	-	-	-
GCMC	0.151	0.182	0.463	0.234	0.244	0.491	-	-	-	-	-	-	-	-	-
CSE	0.211	0.234	0.483	0.26	0.296	0.55	**0.54**	**0.555**	0.753	0.317	0.343	0.58	-	-	-
LCFN	0.188	0.21	0.493	0.22	0.231	0.454	-	-	-	-	-	-	-	-	-
LR-GCCF	0.168	0.214	0.493	0.166	0.168	0.392	0.161	0.174	0.359	-	-	-	-	-	-
SCF	0.196	0.222	0.496	0.165	0.171	0.372	-	-	-	-	-	-	-	-	-

Table 5: Link prediction performance.

Method	Wikipedia		Pinterest		Yelp		MIND		Orkut	
	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR
GEBE^p	**0.964**	0.965	**0.728**	**0.701**	**0.799**	**0.803**	**0.959**	**0.956**	**0.95**	**0.958**
GEBE (Poisson)	0.959	0.964	0.722	0.698	0.796	0.799	0.951	0.949	**0.95**	**0.958**
GEBE (Geometric)	0.949	0.959	0.718	0.696	0.795	0.796	0.944	0.942	0.928	0.934
GEBE (Uniform)	0.951	0.96	0.715	0.697	0.794	0.796	0.942	0.941	0.928	0.932
MHP-BNE	0.942	0.955	0.696	0.656	0.761	0.768	-	-	-	-
MHS-BNE	**0.964**	**0.967**	0.715	0.698	0.786	0.796	-	-	-	-
BiNE	0.956	0.964	0.704	0.647	0.797	0.801	-	-	-	-
BiGI	0.954	0.96	0.697	0.641	0.762	0.754	-	-	-	-
DeepWalk	0.899	0.84	0.659	0.563	0.663	0.622	-	-	-	-
node2vec	0.866	0.838	0.66	0.566	0.674	0.643	-	-	-	-
LINE	0.963	0.965	0.713	0.692	0.796	0.801	0.885	0.907	0.752	0.753
NRP	0.929	0.941	0.712	0.689	0.781	0.785	0.924	0.909	0.911	0.912
BPR	0.951	0.96	0.7	0.624	0.733	0.711	0.939	0.931	-	-
NCF	0.947	0.952	0.691	0.622	0.749	0.732	0.914	0.911	-	-
NGCF	0.96	0.964	0.703	0.637	0.772	0.764	-	-	-	-
LightGCN	0.955	0.964	0.661	0.613	0.668	0.682	-	-	-	-
GCMC	0.948	0.945	0.712	0.689	0.752	0.756	-	-	-	-
CSE	0.944	0.954	0.719	0.656	0.761	0.759	0.949	0.947	0.858	0.856
LCFN	0.922	0.932	0.707	0.682	0.687	0.692	-	-	-	-
LR-GCCF	0.865	0.872	0.658	0.551	0.673	0.581	0.919	0.89	-	-
SCF	0.936	0.944	0.719	0.697	0.748	0.753	-	-	-	-

in existing BNE studies for evaluation [7, 24]. We also follow this setting, and conduct top-N recommendation on weighted bipartite graphs, including DBLP, MovieLens, Last.fm, Netflix and MAG.

For each dataset, we randomly sample 60% edges as the training data, and use the remaining 40% edges as the ground-truth for testing. To ensure the quality of the dataset, we apply the 10-core setting in [29], *i.e.*, retaining users and items with at least ten edges. For each user, we generate his/her top-N ground-truth list by ranking the user u_i's neighbors based on the edge weights to these neighbors (*e.g.*, ratings in user-movie rating graph). Then in top-N recommendation, we follow prior work and adopt the dot product $\mathbf{U}[u_i] \cdot \mathbf{V}[v_j]$ to estimate user u_i's preference on each item v_j, with the requirement that edge (u_i, v_j) is not in training data, and select the top-N ranked items as the top-N recommendation list to u_i. Comparing the top-N recommendation list with the ground-truth list per user, we evaluate the recommendation performance by three popular metrics: *F1*, *Normalized Discounted Cumulative Gain (NDCG)*, and *Mean Reciprocal Rank (MRR)*. For each metric, we report the average score for all users.

Table 4 reports the top-N ($N = 10$) recommendation performance of our proposed methods GEBE^p, GEBE (Uniform), GEBE (Geometric), GEBE (Poisson), and all competitors on five weighted bipartite graphs. The metric scores are all the higher the better. The highest scores are highlighted in bold, while the second best and the third best results are double-underlined and single-underlined, respectively. Observe that GEBE^p consistently outperforms all existing methods by almost all metrics on all datasets, except Last.fm, and the three versions of GEBE, especially GEBE (Poisson), also have high accuracy for top-N recommendation. Note that, for large bipartite graphs including Last.fm, Netflix and MAG, most existing solutions fail to terminate within three days or run out of RAM, and thus their top-N recommendation performances are not reported. On the largest MAG dataset, GEBE^p achieves 26.5% F1, 28.6% NDCG, and 46.8% MRR, outperforming the only competitor that can finish in three days by a significant margin of up to 4.3% for F1, up to 5.6% for NDCG as well as up to 3.2% for MRR. The superiority of our methods is achieved by our unified BNE objective that preserves both MHS and MHP with the consideration of multi-hop paths of all node pairs, and our efficient solvers with approximation guarantee devised in Section 4 and 5. Note that on Last.fm, although competitor CSE has slightly better recommendation performance than GEBE^p on F1 and NDCG metrics, GEBE^p has higher MRR than CSE and is significantly more efficient than CSE, as shown in Figure 2(g), where GEBE^p takes 128 seconds to return results, 98× faster than CSE that costs 3.5 hours. Observe that GEBE (Poisson) is better than GEBE (Geometric) and GEBE (Uniform) in most cases, motivating our specialized design of GEBE^p for Poisson-based BNE computation in Section 5. As shown in Table 4 and Figure 2, GEBE^p is consistently better and faster than GEBE (Poisson), validating the efficacy of our techniques designed in Section 5. In summary, our methods yield superior performance in terms of both recommendation accuracy and efficiency. In Table 4, observe that GEBE^p and GEBE consistently outperform MHP-BNE and MHS-BNE, which demonstrates the benefit of preserving both MHS and MHP simultaneously. In addition, MHP-BNE yields remarkably better accuracy than MHS-BNE, indicating that MHP is more important than MHS in recommendation tasks. We also vary $N = 1, 5, 20$ and 30 and the results are reported in Appendix B of our technical report [73]. Our methods still yield superior performance when $N = 1, 5, 20, 30$, which is consistent with the results when $N = 10$ reported here.

6.4 Link Prediction

Given a pair of nodes, link prediction aims to predict whether it is most likely to form an edge between nodes or not. Following the settings in prior work [7, 24, 26], we regard the link prediction task as a binary classification problem, and evaluate this task on all unweighted bipartite graph datasets, including Wikipedia, Pinterest, Yelp, MIND, and Orkut. Specifically, for each dataset, we randomly remove 40% edges in the input graph G, obtaining a residue graph G' and a set of removed edges. We then randomly sample the same amount of non-existing edges as negative edges. The test set contains both the removed edges and the negative edges. We run all methods on the residual graph (training data) G' to produce embedding vectors $\mathbf{U}[u_i]$ of each node $u_i \in U$ and $\mathbf{V}[v_j]$ of each node $v_j \in V$, and then evaluate the link prediction performance on the test set. For each edge (u_i, v_j) in the training data, we concatenate $\mathbf{U}[u_i]$ and $\mathbf{V}[v_j]$ together as the length-$2k$ feature vector of (u_i, v_j), which is then used to train a binary logistic regression classifier. Finally for every node pair (u_i, v_j) in the testing set, we employ the classifier with the feature vector of (u_i, v_j) as input to predict whether the node pair has class label 1 or 0, *i.e.*, whether there is an edge connecting u_i and v_j or not. We adopt two classical metrics, area under the ROC curve (AUC-ROC) and Precison-Recall curve (AUC-PR), to evaluate the link prediction performance.

Table 5 presents the link prediction performance of our proposed GEBE^p, GEBE (Poisson), GEBE (Geometric), GEBE (Uniform) and all competitors on all five datasets. Note that the highest scores are highlighted in bold, while the second best and the third best results are double-underlined and single-underlined, respectively. Observe that GEBE^p is consistently better than all competitors on all datasets based on both metrics, and GEBE methods also achieve high performance in most cases. In particular, on Pinterest dataset, GEBE^p achieves 72.8% AUC-ROC and 70.1% AUC-PR, while the best competitor NRP has 71.2% AUC-ROC and 68.9% AUC-PR, respectively. On large datasets, including MIND and Orkut, most competitors fail to finish training embedding results within three days, and thus their link prediction performances are not reported. For instance, on Orkut, only three competitors LINE, NRP, and CSE survive, and yield inferior link prediction performance compared with our methods. Specifically, both GEBE^p and GEBE (Poisson) achieve 95% AUC-ROC and 95.8% AUC-PR, improving up to 3.9% and 4.6% compared with NRP with 91.1% AUC-ROC and 91.2% AUC-PR. Also observe that, among the three versions of GEBE, GEBE (Poisson) always achieves the best link prediction performance, which motivates us to design a dedicated solution GEBE^p for our Poisson-based BNE objective in Section 5. In particular, in Theorem 5.1, we prove that GEBE^p in Algorithm 2 provides strong approximation guarantee to solve our Poisson-based BNE objective in Eq. (9), which is demonstrated by the superior performance of GEBE^p shown in Table 5, especially when compared against GEBE (Poisson). To sum up, our proposed methods achieve high performance in terms of both link prediction accuracy and computational efficiency, considering the efficiency evaluation reported in Section 6.2. In Table 5, GEBE^p and GEBE outperform MHS-BNE and MHP-BNE for link prediction tasks, validating the effectiveness of preserving both MHP and MHS measures at the same time. Additionally, MHS-BNE considerably outperforms MHP-BNE in link predictions, which indicates that MHS is more important for link predictions than MHP.

6.5 Parameter Analysis

In this section, we further study the effect of varying the parameters in GEBE^p, including λ used in Eq. (8) and the error threshold ϵ used in SVD at Line 1 of Algorithm 2, as well as parameter τ in GEBE (Poisson). We conduct the evaluation on DBLP, MovieLens, Last.fm, and Netflix datasets for top-N=10 recommendation

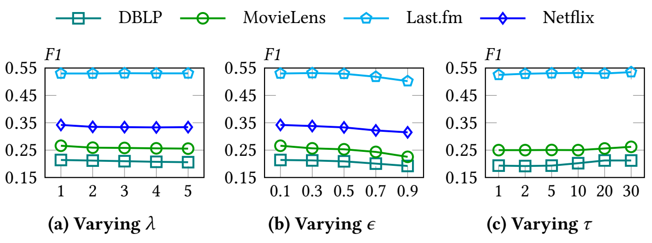

Figure 4: Top-10 recommendation when varying λ, ϵ and τ.

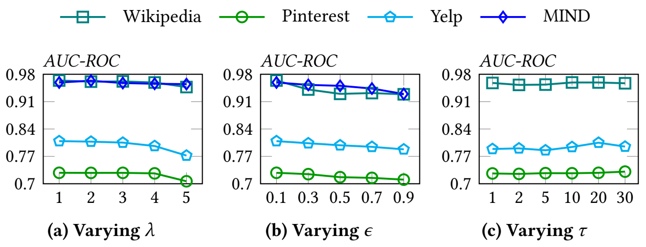

Figure 5: Link prediction when varying λ, ϵ and τ.

task, and on Wikipedia, Pinterest, Yelp and MIND datasets for link prediction task. In particular, we vary $\lambda \in \{1, 2, 3, 4, 5\}$ and $\epsilon \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ in GEBE^p, and $\tau \in \{1, 2, 5, 10, 20, 30\}$ in GEBE (Poisson). When λ increases, it means that we assign more importance to paths with long lengths. When ϵ is small, we get more accurate SVD results. As τ increases, GEBE (Poisson) returns more accurate embeddings.

Figure 4(a) reports the top-10 recommendation F1 scores of GEBE^p when we vary λ from 1 to 5. Observe that the F1 scores are quite stable with a slight decreasing when λ increases. Similar observation can be made from Figure 5(a), which reports the link prediction AUC-ROC scores when varying λ. The observations indicate that shorter paths should play an important role in bipartite network embedding. Therefore, we choose $\lambda = 1$ in our experiments.

Figure 4(b) reports the top-10 recommendation F1 scores of GEBE^p when varying ϵ. We observe that the F1 scores are decreasing when we allow a larger error threshold ϵ in SVD. Similarly, in Figure 5(b) that presents the link prediction AUC-ROC scores when varying ϵ, the AUC-ROC scores also drop as ϵ increases. We conclude that accurate SVD at Line 1 of Algorithm 2 can improve performance, and thus we choose $\epsilon = 0.1$ in our experiments.

Figure 4(c) shows the F1 scores of GEBE for top-10 recommendation on DBLP, MovieLens, and Last.fm. The F1 scores increase slightly as τ increases from 1 to 30. Figure 5(c) plots the AUC-ROC results of GEBE for link prediction on Wikipedia, Pinterest, and Yelp. Observe that the prediction accuracy of GEBE does not vary significantly when τ changes.

7 RELATED WORK

Bipartite Network Embedding. BiNE [24] performs a large number of biased random walks to preserve the long-tail distribution of nodes, and obtains bipartite embeddings by considering both edges and paths in the input graph. BiGI [7] captures global properties including community structures of homogeneous nodes and long-range dependencies of heterogeneous nodes, and solves a local-global infomax objective using multilayer perceptrons (MLPs) to produce BNE results. These BNE methods incur immense overheads in either conducting numerous random walks or relying on expensive MLPs, and are not scalable to massive graphs, as demonstrated in our experiments.

Homogeneous Network Embedding. A simple way to solve BNE problem is to regard bipartite graphs as typeless homogeneous graphs and then apply existing homogeneous network embedding (HONE) solutions to get embedding results. There exists a large collection of HONE solutions, *e.g.*, [10, 23, 26, 50, 60, 61, 88]. For instance, DeepWalk [50], node2vec [26] and LINE [60] learn embeddings based on Skip-gram model [45] with random walks sampled on homogeneous graphs. Another line of research employs deep learning techniques for HONE computation, which are computationally expensive over massive graphs [1, 8, 15, 16, 39, 62–64, 79, 89]. Further, recent studies develop efficient matrix factorization based techniques for scalable HONE computation, *e.g.*, [43, 48, 51, 52, 52, 72, 74, 75, 82, 84, 85]. However, a major issue of all these HONE solutions is that they are not designed for bipartite graphs, and thus ignore the unique characteristics and type information of bipartite graphs, resulting to inferior BNE result quality, as validated in our experiments in Section 6.

Heterogeneous Network Embedding. A heterogeneous graph contains multiple types of nodes and/or edges. In literature, many heterogeneous Network Embedding (HENE) methods are proposed, *e.g.*, [19, 57, 69, 80]. Specifically, Metapath2vec [18] proposes meta-path based random walks to build the heterogeneous neighborhoods of nodes, and then leverages a heterogeneous Skip-gram model [44, 45] to learn HENE results. HERec [57] also samples meta-path based random walks, and then learn embeddings by jointly optimizing an extended matrix factorization model together with a set of customized fusion functions. HAN [69] learns embeddings by aggregating features from meta-path-based neighbors via a heterogeneous graph neural network with node-level and semantic-level attentions. GTN [80] transforms a heterogeneous graph into multiple graphs defined by meta-paths with arbitrary edge types and lengths, and then learns embeddings via convolution on these multiple graphs. Although these HENE methods are able to solve BNE problem by simply regarding bipartite graphs as heterogeneous graphs, HENE methods yield sub-optimal BNE performance as demonstrated in [7], since HENE methods largely rely on handcrafted meta-paths, and are not specialized to utilize the unique features of bipartite graphs.

Collaborative Filtering. Collaborative Filtering (CF) [56] is well-studied in recommender systems [13, 20, 68]. CF is a methodology of making predictions to users based on user preferences, and is widely applied on user-item bipartite graphs. A common CF paradigm is to obtain user and item embeddings and then perform recommendations. In other words, many CF methods can be applied for BNE computation. Matrix-factorization-based CF methods [32, 35, 37, 38, 53, 59] decompose a user-item interaction matrix to obtain embedding results. For instance, BPR [53] method performs the matrix factorization with a pairwise ranking-aware objective. Recently, many studies start to exploit deep learning techniques to capture the nonlinear features of user-item interactions, to facilitate effective CF recommendation, *e.g.*, [28, 31, 33, 40, 68, 70]. NCF [31] employs non-linear neural networks to model the user-item interactions. GCMC [5] applies a one-layer graph neural network (GNN) [36] on the user-item graph to model the direct connections between users and items. SCF [87] contains a spectral convolution operation to discover all possible connections between users and items in spectral domain. NGCF [67] learns embeddings via a multi-layer GNN that captures the high-order connectivities between users and items. CSE [9] considers both direct user-item relations, as well as high-order neighborhood proximity in a joint-learning model by sampling k-order random walks. LightGCN [30] first learns initial embeddings, and then propagates the embeddings on the input graph to obtain the final embeddings. IGMC [83] trains a GNN preserving local graph patterns around a user-item pair. LCFN [78] removes noise in the observed data, and then reduces the complexity of graph convolution in an unscathed way for recommendation. LR-GCCF [11] proposes a residual network structure that is specifically designed for CF to alleviate the over-smoothing problem in GNN. Note that all these CF-based methods are either inefficient or ineffective to output high-quality BNE results, or both, as evaluated in our experiments.

8 CONCLUSION

This paper presents GEBE, a generic bipartite network embedding framework with a specialized algorithm GEBE^p, which scales to massive graphs with billions of edges and obtains state-of-the-art predictive accuracy in downstream tasks. In particular, we develop two generic measures to capture the multi-hop similarity (MHS) between same-type nodes and multi-hop proximity (MHP) between heterogeneous nodes, propose a unified BNE objective preserving all MHS and MHP scores with three possible instantiations, and develop a series of highly efficient techniques based on rigorous analysis and sophisticated algorithm designs. Extensive experiments show that our methods, especially GEBE^p, achieve substantial improvement over state-of-the-art solutions in terms of both efficiency and effectiveness. As for future work, we intend to extend our solutions to handle bipartite attributed graphs by augmenting the network embeddings with raw/processed attributes.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. This work was supported by the the Ministry of Education Singapore (Number MOE2018-T2-2-091). Jieming Shi is supported by Hong Kong RGC ECS (No. 25201221) and PolyU Start-up Fund (P0033898). Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies and organizations.

A PROOFS

Proof of Lemma 2.1. If there are no paths from u_i to u_l, we have $\mathbf{H}[u_i,u_l]=0$; hence $s(u_i,u_l)=0$. In addition, $s(u_i,u_i)=\frac{\mathbf{H}[u_i,u_i]}{\mathbf{H}[u_i,u_i]}=1$. Note that all edge weights in E are non-negative, meaning that $\forall u_i,u_l\in U, s(u_i,u_l)\ge 0$. We then prove that, for any $0\le \ell\le\tau$ and nodes $u_i,u_l\in U$, we have

$$(\mathbf{W}\mathbf{W}^\top)^\ell[u_i,u_l]\le\sqrt{(\mathbf{W}\mathbf{W}^\top)^\ell[u_i,u_i]}\cdot\sqrt{(\mathbf{W}\mathbf{W}^\top)^\ell[u_l,u_l]}. \tag{18}$$

First, if $\ell=0$, it is obvious that

$$\frac{(\mathbf{W}\mathbf{W}^\top)^0[u_i,u_l]}{\sqrt{(\mathbf{W}\mathbf{W}^\top)^0[u_i,u_i]}\cdot\sqrt{(\mathbf{W}\mathbf{W}^\top)^0[u_l,u_l]}}\le 1.$$

If ℓ is an odd integer, by Cauchy–Schwarz inequality, we have

$$\frac{(\mathbf{W}\mathbf{W}^\top)^\ell[u_i,u_l]}{\sqrt{(\mathbf{W}\mathbf{W}^\top)^\ell[u_i,u_i]}\cdot\sqrt{(\mathbf{W}\mathbf{W}^\top)^0[u_l,u_l]}}$$
$$=\frac{((\mathbf{W}\mathbf{W}^\top)^{\frac{\ell-1}{2}}\mathbf{W})[u_i]\cdot((\mathbf{W}\mathbf{W}^\top)^{\frac{\ell-1}{2}}\mathbf{W})[u_l]}{\|((\mathbf{W}\mathbf{W}^\top)^{\frac{\ell-1}{2}}\mathbf{W})[u_i]\|_2\cdot\|((\mathbf{W}\mathbf{W}^\top)^{\frac{\ell-1}{2}}\mathbf{W})[u_l]\|_2}\le 1.$$

Similarly, if ℓ is an even integer and $\ell>0$, we obtain

$$\frac{(\mathbf{W}\mathbf{W}^\top)^\ell[u_i,u_l]}{\sqrt{(\mathbf{W}\mathbf{W}^\top)^\ell[u_i,u_i]}\cdot\sqrt{(\mathbf{W}\mathbf{W}^\top)^0[u_l,u_l]}}$$
$$=\frac{(\mathbf{W}\mathbf{W}^\top)^{\frac{\ell}{2}}[u_i]\cdot(\mathbf{W}\mathbf{W}^\top)^{\frac{\ell}{2}}[u_l]^\top}{\|(\mathbf{W}\mathbf{W}^\top)^{\frac{\ell}{2}}[u_i]\|_2\cdot\|(\mathbf{W}\mathbf{W}^\top)^{\frac{\ell}{2}}[u_l]\|_2},$$

which is bounded by 1 by Cauchy–Schwarz inequality. Hence, Eq. (18) holds for any $\ell>0$ and nodes $u_i,u_l\in U$.

According to Eq. (3) and Cauchy–Schwarz inequality, we obtain

$$\mathbf{H}[u_i,u_l]=\sum_{\ell=0}^{\tau}\omega(\ell)\cdot(\mathbf{W}\mathbf{W}^\top)^\ell[u_i,u_l]$$
$$\le\sum_{\ell=0}^{\tau}\omega(\ell)\sqrt{(\mathbf{W}\mathbf{W}^\top)^\ell[u_i,u_i]}\sqrt{(\mathbf{W}\mathbf{W}^\top)^\ell[u_l,u_l]}$$
$$\le\sqrt{\sum_{\ell=0}^{\tau}\omega(\ell)(\mathbf{W}\mathbf{W}^\top)^\ell[u_i,u_i]}\sqrt{\sum_{\ell=0}^{\tau}\omega(\ell)(\mathbf{W}\mathbf{W}^\top)^\ell[u_l,u_l]},$$

leading to $\mathbf{H}[u_i,u_l]\le\sqrt{\mathbf{H}[u_i,u_i]}\cdot\sqrt{\mathbf{H}[u_l,u_l]}$ and $s(u_i,u_l)\le 1$. Therefore, the lemma is proved. □

Proof of Lemma 2.2. Recall that for any two unit vectors $\mathbf{u},\mathbf{v}\in\mathbb{R}^{n\times k}$, $\|\mathbf{u}-\mathbf{v}\|_2^2=2(1-\cos(\mathbf{u},\mathbf{v}))$ holds. Thus, we obtain

$$\frac{1}{2}\left\|\frac{\mathbf{U}[u_i]}{\|\mathbf{U}[u_i]\|_2}-\frac{\mathbf{U}[u_l]}{\|\mathbf{U}[u_l]\|_2}\right\|_2^2=1-\frac{\mathbf{U}[u_i]}{\|\mathbf{U}[u_i]\|_2}\cdot\frac{\mathbf{U}[u_l]}{\|\mathbf{U}[u_l]\|_2} \tag{19}$$

Since $\mathcal{L}=0$, we have $\mathbf{U}[u_i]\cdot\mathbf{V}[v_j]=\mathbf{P}[u_i,v_j]$ for every node pair $(u_i,v_j)\in U\times V$ and

$$\frac{\mathbf{U}[u_i]}{\|\mathbf{U}[u_i]\|_2}\cdot\frac{\mathbf{U}[u_l]}{\|\mathbf{U}[u_l]\|_2}=s(u_i,u_l)$$

for every node pair $(u_i,u_l)\in U\times U$. Recall that $\mathbf{P}[u_i,v_j]=(\mathbf{H}\mathbf{W})[u_i,v_j]$. Then, we get

$$\mathbf{U}[u_i]\cdot\mathbf{V}[v_j]=\sum_{u_l\in U}s(u_i,u_l)\cdot w(u_l,v_j)\cdot\sqrt{\mathbf{H}[u_i,u_i]}\cdot\sqrt{\mathbf{H}[u_l,u_l]},$$

which leads to

$$\mathbf{V}[v_j]=\frac{\sqrt{\mathbf{H}[u_i,u_i]}}{\|\mathbf{U}[u_i]\|_2}\sum_{u_l\in U}\frac{\mathbf{U}[u_l]}{\|\mathbf{U}[u_l]\|_2}\cdot w(u_l,v_j)\cdot\sqrt{\mathbf{H}[u_l,u_l]}.$$

Thus, for every two nodes $v_i,v_h\in V$, we have

$$\mathbf{V}[v_j]\cdot\mathbf{V}[v_h]=\frac{\mathbf{H}[u_i,u_i]}{\|\mathbf{U}[u_i]\|_2^2}\sum_{u_l,u_x\in U}w(u_l,v_j)\cdot\mathbf{H}[u_l,u_x]\cdot w(u_x,v_h)$$
$$=\frac{\mathbf{H}[u_i,u_i]}{\|\mathbf{U}[u_i]\|_2^2}\cdot\sum_{u_l\in N(v_j)}w(u_l,v_j)\cdot\mathbf{P}[u_l,v_h]. \tag{20}$$

Based on Eq. (20), for any two nodes $v_j, v_h \in V$, we obtain

$$\begin{aligned}
&\frac{1}{2}\left\|\frac{\mathbf{V}[v_j]}{\|\mathbf{V}[v_j]\|_2} - \frac{\mathbf{V}[v_h]}{\|\mathbf{V}[v_h]\|_2}\right\|_2^2 = 1 - \frac{\mathbf{V}[v_j]}{\|\mathbf{V}[v_j]\|_2} \cdot \frac{\mathbf{V}[v_h]}{\|\mathbf{V}[v_h]\|_2} \\
&= 1 - \frac{\sum_{u_l \in N(v_j)} w(v_j, u_l) \cdot \mathbf{P}[u_l, v_h]}{\sqrt{\sum_{u_l \in N(v_j)} w(v_j, u_l)\mathbf{P}[u_l, v_j]}\sqrt{\sum_{u_l \in N(v_h)} w(v_h, u_l)\mathbf{P}[u_l, v_h]}} \\
&= 1 - \frac{\sum_{\ell=1}^{\tau} \omega(\ell) \cdot (\mathbf{W}^\top\mathbf{W})^\ell[v_j, v_h]}{\sqrt{\sum_{\ell=1}^{\tau} \omega(\ell) \cdot (\mathbf{W}^\top\mathbf{W})^\ell[v_j, v_j]} \cdot \sqrt{\sum_{\ell=1}^{\tau} \omega(\ell) \cdot (\mathbf{W}^\top\mathbf{W})^\ell[v_h, v_h]}} \\
&= 1 - s(v_j, v_h).
\end{aligned}$$

The lemma follows. □

Proof of Theorem 3.1. We need the following theorem.

Theorem A.1 (Eckart–Young Theorem [25]). *Suppose that $\mathbf{M}_k \in \mathbb{R}^{n \times k}$ is the rank-k approximation to $\mathbf{M} \in \mathbb{R}^{n \times n}$ obtained by exact SVD, then*

$$\min_{rank(\widehat{\mathbf{M}}) \le k} \|\mathbf{M} - \widehat{\mathbf{M}}\|_2 = \|\mathbf{M} - \mathbf{M}_k\|_2 = \sigma_{k+1},$$

where σ_i represents the i-th largest singular value of $\mathbf{M}$.

Suppose that $\mathbf{X}, \mathbf{Y}$ are the optimal embeddings to our objective in Eq. (9) as defined in Eq. (10), we have $\mathcal{L}(\mathbf{X}, \mathbf{Y}) = 0$ and $\mathbf{X}\mathbf{X}^\top = \mathbf{H}$. Since $\mathbf{U}^*, \mathbf{V}^*$ are defined as in Eq .(13), by Eckart–Young Theorem (*i.e.*, Theorem A.1) and Lemma 4.1 in [86], we then have

$$\|\mathbf{U}^*\mathbf{U}^{*\top} - \mathbf{X}\mathbf{X}^\top\|_2 = \sigma_{k+1}, \tag{21}$$

where σ_i represents the i-th largest singular value of $\mathbf{H}$. In addition, Eq. (21) implies that $\mathbf{U}^*\mathbf{U}^{*\top}$ is the best rank-k approximation of $\mathbf{H}$ both in the Frobenius norm and in the spectral norm.

We define the following two loss functions

$$\mathcal{L}_p(\mathbf{U}^*, \mathbf{V}^*) = \sum_{u_i \in U}\sum_{v_j \in V} \left(\mathbf{U}^*[u_i] \cdot \mathbf{V}^*[v_j] - \mathbf{P}[u_i, v_j]\right)^2,$$

$$\mathcal{L}_s(\mathbf{U}^*) = \sum_{u_i, u_l \in U} \left(\frac{\mathbf{U}^*[u_i]}{\|\mathbf{U}^*[u_i]\|_2} \cdot \frac{\mathbf{U}^*[u_l]}{\|\mathbf{U}^*[u_l]\|_2} - s(u_i, u_l)\right)^2.$$

Therefore, by Eq. (19), $\mathcal{L}(\mathbf{U}^*, \mathbf{V}^*) = \mathcal{L}_p(\mathbf{U}^*, \mathbf{V}^*) + \mathcal{L}_s(\mathbf{U}^*)$.

We first consider the loss $\mathcal{L}_p(\mathbf{U}^*, \mathbf{V}^*)$. According to Eq. (21) and the *spectral submultiplicativity*, *e.g.*, $\|\mathbf{M}\mathbf{B}\|_F \le \|\mathbf{M}\|_2 \cdot \|\mathbf{B}\|_F$ for any two matrices $\mathbf{M}, \mathbf{B}$, we obtain

$$\begin{aligned}
\mathcal{L}_p(\mathbf{U}^*, \mathbf{V}^*) - \mathcal{L}_p(\mathbf{X}, \mathbf{Y}) &= \frac{1}{|U| \cdot |V|} \|\mathbf{U}^*\mathbf{V}^{*\top} - \mathbf{X}\mathbf{Y}^\top\|_F^2 \\
&= \frac{1}{|U| \cdot |V|} \|\mathbf{U}^*\mathbf{U}^{*\top}\mathbf{W} - \mathbf{X}\mathbf{X}^\top\mathbf{W}\|_F^2 \\
&\le \frac{1}{|U| \cdot |V|} \|\mathbf{U}^*\mathbf{U}^{*\top} - \mathbf{X}\mathbf{X}^\top\|_2^2 \cdot \|\mathbf{W}\|_F^2 \\
&\le \frac{\sigma_{k+1}^2}{|U| \cdot |V|} \cdot \sum_{(u_i, v_j) \in E} w(u_i, v_j)^2.
\end{aligned} \tag{22}$$

Next, we consider the loss $\mathcal{L}_s(\mathbf{U}^*, \mathbf{V}^*)$.

$$\begin{aligned}
&\mathcal{L}_s(\mathbf{U}^*) - \mathcal{L}_s(\mathbf{X}) \\
&= \frac{2}{|U|^2} \sum_{u_i, u_l \in U} \left(\frac{\mathbf{U}^*[u_i] \cdot \mathbf{U}^*[u_l]}{\|\mathbf{U}^*[u_i]\|_2 \cdot \|\mathbf{U}^*[u_l]\|_2} - \frac{\mathbf{X}[u_i] \cdot \mathbf{X}[u_l]}{\|\mathbf{X}[u_i]\|_2 \cdot \|\mathbf{X}[u_l]\|_2}\right)^2.
\end{aligned} \tag{23}$$

Let $\mathbf{Z}, \mathbf{\Lambda}$ be the exact full eigenvectors and eigenvalues of $\mathbf{H}$. By Lemma 4.1 in [86] and and the property of eigenvectors, *i.e.*, $\mathbf{Z}^\top\mathbf{Z} = \mathbf{I}$, for any node $u_i \in U$, we have

$$0 \le \|\mathbf{X}[u_i]\|_2^2 - \|\mathbf{U}^*[u_i]\|_2^2 = \sum_{j=k+1}^{|V|} \mathbf{Z}[u_i, j]^2 \cdot \mathbf{\Lambda}[j, j] \le \sigma_{k+1}.$$

Let $\mathbf{D}$ and $\widehat{\mathbf{D}}$ be two $|U| \times |U|$ diagonal matrices, where each diagonal entry

$$\mathbf{D}[u_i, u_l] = \frac{1}{\|\mathbf{X}[u_i]\|_2},\ \widehat{\mathbf{D}}[u_i, u_l] = \frac{1}{\|\mathbf{U}^*[u_i]\|_2},$$

respectively. Then, Eq. (23) becomes

$$\begin{aligned}
\mathcal{L}_s(\mathbf{U}^*) - \mathcal{L}_s(\mathbf{X}) &= \frac{2}{|U|^2} \|\widehat{\mathbf{D}}\mathbf{U}^*\mathbf{U}^{*\top}\widehat{\mathbf{D}} - \mathbf{D}\mathbf{X}\mathbf{X}^\top\mathbf{D}\|_F^2 \\
&\le \frac{2}{|U|^2} \|\widehat{\mathbf{D}} \cdot (\mathbf{U}^*\mathbf{U}^{*\top} - \mathbf{X}\mathbf{X}^\top) \cdot \widehat{\mathbf{D}}\|_F^2 \\
&\le \frac{2}{|U|^2} \|\widehat{\mathbf{D}}\|_F^2 \cdot \|\mathbf{U}^*\mathbf{U}^{*\top} - \mathbf{X}\mathbf{X}^\top\|_2^2 \\
&\le \frac{\sigma_{k+1}^2}{|U|^2} \cdot \sum_{u_i \in U} \frac{2}{(\mathbf{H}[u_i, u_i] - \sigma_{k+1})^2}.
\end{aligned} \tag{24}$$

Combining Equations (22) and (24) establishes the theorem. □

Proof of Theorem 4.1. According to Lines 3-6 in Algorithm 1, we obtain the result $\mathbf{Q} = \sum_{\ell=0}^{\tau} \omega(\ell) \cdot (\mathbf{W}\mathbf{W}^\top)^\ell \mathbf{Z}'_k = \mathbf{H}\mathbf{Z}'_k$ after τ iterations of the power method. Since $\mathbf{H}$ is symmetric and $\mathbf{Z}'_k$ converges within t Krylov subspace iterations, the columns of $\mathbf{Z}'_k$ contains the exact top-k eigenvectors of $\mathbf{H}$, *i.e.*, $\mathbf{Z}'_k = \mathbf{Z}_k$ and the diagonal entries of $\mathbf{R}$ are the exact top-k eigenvalues of $\mathbf{H}$, *i.e.*, $\mathbf{\Lambda}'_k = \mathbf{\Lambda}_k$ [55]. Therefore, we obtain $\mathbf{U} = \mathbf{Z}'_k\sqrt{\mathbf{\Lambda}'_k} = \mathbf{Z}_k\sqrt{\mathbf{\Lambda}_k} = \mathbf{U}^*$ and $\mathbf{V} = \mathbf{W}^\top\mathbf{Z}_k\sqrt{\mathbf{\Lambda}_k} = \mathbf{W}^\top\mathbf{U}^* = \mathbf{V}^*$, which proves the lemma. □

Proof of Theorem 5.1. Denote by σ_{k+1} the $(k+1)$-th largest singular value of $\mathbf{W}$. At Line 1 in Algorithm 2, Σ'_k is got by performing RandomizedSVD over $\mathbf{W}$. By Theorem 1 in [47], we have

$$\forall 1 \le i \le k,\ \left|\Sigma'_k[i, i]^2 - \sigma_i^2\right| \le \epsilon \cdot \sigma_{k+1}^2 \tag{25}$$

Let $\mathbf{\Phi}_k, \Sigma_k$ be the *exact* top-k left singular vectors and top-k singular values of $\mathbf{W}$, respectively. According to Lines 1-4 in Algorithm 2, $\mathbf{U}\mathbf{U}^\top = \mathbf{\Phi}'_k e^{-\lambda} e^{\lambda\Sigma_k'^2} \mathbf{\Phi}'^{\top}_k$. Lemma 3.1 and Eq. (17) yield

$$\|\mathbf{U}_\lambda^*\mathbf{U}_\lambda^{*\top} - \mathbf{U}\mathbf{U}^\top\|_F^2 = \|\mathbf{\Phi}_k \frac{e^{\lambda\Sigma_k^2}}{e^\lambda} \mathbf{\Phi}_k^\top - \mathbf{\Phi}'_k \frac{e^{\lambda\Sigma_k'^2}}{e^\lambda} \mathbf{\Phi}'^{\top}_k\|_F^2$$

Then, according to Definition 8 in [47], we have

$$\|\mathbf{U}_\lambda^*\mathbf{U}_\lambda^{*\top} - \mathbf{U}\mathbf{U}^\top\|_F^2 = \|\mathbf{\Phi}_k \frac{e^{\lambda\Sigma_k^2}}{e^\lambda} \mathbf{\Psi}_k^\top\|_F^2 - \|\mathbf{\Phi}'_k \frac{e^{\lambda\Sigma_k'^2}}{e^\lambda} \mathbf{\Phi}'^{\top}_k\|_F^2.$$

Applying Eq. (25) to the above equation gives the following result:

$$\begin{aligned}
\|\mathbf{U}_\lambda^*\mathbf{U}_\lambda^{*\top} - \mathbf{U}\mathbf{U}^\top\|_F^2 &= \sum_{i=1}^{k} \frac{e^{\lambda\Sigma_k^2[i,i]}}{e^\lambda} - \sum_{i=1}^{k} \frac{e^{\lambda\Sigma_k'^2[i,i]}}{e^\lambda} \\
&\le \sum_{i=1}^{k} \left(\frac{e^{\lambda\sigma_i^2}}{e^\lambda} - \frac{e^{\lambda(\sigma_i^2 - \epsilon\sigma_{k+1}^2)}}{e^\lambda}\right).
\end{aligned}$$

In addition, note that $\mathbf{V}_\lambda^* = \mathbf{W}^\top\mathbf{U}_\lambda^*$ and $\mathbf{V} = \mathbf{W}^\top\mathbf{U}$ (Line 4 in Algorithm 2). By the *spectral submultiplicativity*, we can derive that

$$\begin{aligned}
\|\mathbf{U}_\lambda^*\mathbf{V}_\lambda^* - \mathbf{U}\mathbf{V}\|_F^2 &= \|\mathbf{U}_\lambda^*\mathbf{U}_\lambda^{*\top}\mathbf{W} - \mathbf{U}\mathbf{U}^\top\mathbf{W}\|_F^2 \\
&\le \|\mathbf{U}_\lambda^*\mathbf{U}_\lambda^{*\top} - \mathbf{U}\mathbf{U}^\top\|_F^2 \|\mathbf{W}\|_2^2 \\
&= \|\mathbf{U}_\lambda^*\mathbf{U}_\lambda^{*\top} - \mathbf{U}\mathbf{U}^\top\|_F^2 \cdot \sigma_1^2 \\
&\le \sigma_1^2 \cdot \sum_{i=1}^{k} \left(\frac{e^{\lambda\sigma_i^2}}{e^\lambda} - \frac{e^{\lambda\left(\sigma_i^2 - \epsilon\sigma_{k+1}^2\right)}}{e^\lambda}\right),
\end{aligned}$$

which finishes our proof. □

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alexander A Alemi. 2018. Watch Your Step: Learning Node Embeddings via Graph Attention. In *NeurIPS*. 9180–9190.

[2] Ioannis Antonellis, Hector Garcia Molina, and Chi Chao Chang. 2008. Simrank++ query rewriting through link analysis of the click graph. *Proceedings of the VLDB Endowment* (2008), 408–421.

[3] Albert-László Barabási and Eric Bonabeau. 2003. Scale-free networks. *Scientific american* (2003), 60–69.

[4] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *KDD cup and workshop*. 35.

[5] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[6] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling Graph Neural Networks with Approximate PageRank. In *SIGKDD*.

[7] Jiangxia Cao, Xixun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. 2021. Bipartite Graph Embedding via Mutual Information Maximization. In *WSDM*.

[8] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep Neural Networks for Learning Graph Representations.

[9] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. 2019. Collaborative similarity embedding for recommender systems. In *The WebConf*. 2637–2643.

[10] Hongxu Chen, Hongzhi Yin, Tong Chen, Quoc Viet Hung Nguyen, Wen-Chih Peng, and Xue Li. 2019. Exploiting centrality information with graph convolutions for network representation learning. In *ICDE*. 590–601.

[11] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI*, Vol. 34. 27–34.

[12] Fan Chung. 2007. The heat kernel as the pagerank of a graph. *PNAS* 104, 50 (2007), 19735–19740.

[13] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[14] Haskell B Curry. 1944. The method of steepest descent for non-linear minimization problems. *Quart. Appl. Math.* 2, 3 (1944), 258–261.

[15] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. 2018. Adversarial network embedding. In *AAAI*.

[16] Quanyu Dai, Xiao Shen, Liang Zhang, Qiang Li, and Dan Wang. 2019. Adversarial Training Methods for Network Embedding. In *WWW*. 329–339.

[17] James W Demmel. 1997. *Applied numerical linear algebra*. SIAM.

[18] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*. 135–144.

[19] Yuxiao Dong, Ziniu Hu, Kuansan Wang, Yizhou Sun, and Jie Tang. 2020. Heterogeneous network representation learning. In *IJCAI*. 4861–4867.

[20] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *SIGIR*. 515–524.

[21] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *The WebConf*. 1775–1784.

[22] László Erdős, Antti Knowles, Horng-Tzer Yau, Jun Yin, et al. 2013. Spectral statistics of Erdős–Rényi graphs I: Local semicircle law. *The Annals of Probability* (2013), 2279–2375.

[23] Hongchang Gao and Heng Huang. 2018. Self-paced network embedding. In *KDD*. 1406–1415.

[24] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. BiNE: Bipartite Network Embedding. 715–724.

[25] Gene H Golub and Charles F Van Loan. 1996. Matrix computations. *Johns Hopkins University, Press* (1996).

[26] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. 855–864.

[27] Chaoyang He, Tian Xie, Yu Rong, Wenbing Huang, Junzhou Huang, Xiang Ren, and Cyrus Shahabi. 2019. Cascade-BGNN: Toward Efficient Self-supervised Representation Learning on Large-scale Bipartite Graphs. *arXiv preprint arXiv:1906.11994* (2019).

[28] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *RecSys*. 161–169.

[29] Ruining He and Julian McAuley. 2016. VBPR: visual Bayesian Personalized Ranking from implicit feedback. In *AAAI*. 144–150.

[30] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[31] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[32] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*. 549–558.

[33] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *The WebConf*. 193–201.

[34] Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*. 271–279.

[35] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *SIGKDD*. 659–667.

[36] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[37] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*. 426–434.

[38] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[39] Yi-An Lai, Chin-Chi Hsu, Wen Hao Chen, Mi-Yen Yeh, and Shou-De Lin. 2017. Prune: Preserving proximity and global ranking for network embedding. *NeurIPS* 30 (2017), 5257–5266.

[40] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *The WebConf*. 689–698.

[41] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[42] Boge Liu, Long Yuan, Xuemin Lin, Lu Qin, Wenjie Zhang, and Jingren Zhou. 2019. Efficient (α, β)-Core Computation: An Index-Based Approach. In *The WebConf*. 1130–1141.

[43] Xin Liu, Tsuyoshi Murata, Kyoung-Sook Kim, Chatchawan Kotarasu, and Chenyi Zhuang. 2019. A general view for network embedding as matrix factorization. In *WSDM*. 375–383.

[44] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[45] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *NeurIPS* 26 (2013), 3111–3119.

[46] Cleve Moler and Charles Van Loan. 1978. Nineteen dubious ways to compute the exponential of a matrix. *SIAM review* (1978), 801–836.

[47] Cameron Musco and Christopher Musco. 2015. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. In *NeurIPS*. 1396–1404.

[48] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric Transitivity Preserving Graph Embedding. 1105–1114.

[49] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[50] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. 701–710.

[51] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. In *The WebConf*. 1509–1520.

[52] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. 459–467.

[53] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[54] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature* (1986), 533–536.

[55] Heinz Rutishauser. 1969. Computational aspects of FL Bauer's simultaneous iteration method. *Numer. Math.* (1969), 4–13.

[56] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *The WebConf*. 285–295.

[57] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2018. Heterogeneous information network embedding for recommendation. *TKDE* 31, 2 (2018), 357–370.

[58] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *The WebConf*.

[59] Gábor Takács and Domonkos Tikk. 2012. Alternating least squares for personalized ranking. In *RecSys*. 83–90.

[60] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *The WebConf*. 1067–1077.

[61] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. VERSE: Versatile Graph Embeddings from Similarity Measures. In *The WebConf*. 539–548.

[62] Ke Tu, Peng Cui, Xiao Wang, Philip S Yu, and Wenwu Zhu. 2018. Deep recursive network embedding with regular equivalence. In *KDD*. 2357–2366.

[63] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. 1225–1234.

[64] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xie Xing, and Minyi Guo. 2018. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. In *AAAI*.

[65] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *SIGKDD*. 839–848.

[66] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2019. Vertex Priority Based Butterfly Counting for Large-Scale Bipartite Networks. *PVLDB* 12, 10 (2019), 1139–1152.

[67] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[68] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*. 153–162.

[69] Wang Xiao, Ji Houye, Shi Chuan, Wang Bai, Cui Peng, Yu P., and Ye Yanfang. 2019. Heterogeneous Graph Attention Network. *The WebConf* (2019).

[70] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems.. In *IJCAI*, Vol. 17. 3203–3209.

[71] Yoshihiro Yamanishi, Masaaki Kotera, Minoru Kanehisa, and Susumu Goto. 2010. Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework. *Bioinformatics* (2010), 246–254.

[72] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. 2017. Fast network embedding enhancement via high order proximity approximation.. In *IJCAI*. 3894–3900.

[73] Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. 2021. Technical Report. https://sites.google.com/view/gebe-technical-report.

[74] Renchi Yang, Jieming Shi, Xiaokui Xiao, Yin Yang, and Sourav S Bhowmick. 2020. Homogeneous network embedding for massive graphs via reweighted personalized PageRank. *PVLDB* 13, 5 (2020), 670–683.

[75] Yuan Yin and Zhewei Wei. 2019. Scalable Graph Embeddings via Sparse Transpose Proximities.

[76] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*. 974–983.

[77] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*. 974–983.

[78] Wenhui Yu and Zheng Qin. 2020. Graph Convolutional Network for Recommendation with Low-pass Collaborative Filters. In *ICML*. PMLR, 10936–10945.

[79] Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. 2018. Learning deep network representations with adversarially regularized autoencoders. In *SIGKDD*. 2663–2671.

[80] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph Transformer Networks. In *NeurIPS*. 11960–11970.

[81] Chuxu Zhang, Lu Yu, Xiangliang Zhang, and Nitesh V Chawla. 2018. Task-guided and semantic-aware ranking for academic author-paper correlation inference. In *IJCAI*. 3641–3647.

[82] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: Fast and Scalable Network Representation Learning. In *IJCAI*. 4278–4284.

[83] Muhan Zhang and Yixin Chen. 2020. Inductive Matrix Completion Based on Graph Neural Networks. In *ICLR*.

[84] Ziwei Zhang, Peng Cui, Haoyang Li, Xiao Wang, and Wenwu Zhu. 2018. Billion-scale Network Embedding with Iterative Random Projection. In *ICDM*. 787–796.

[85] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-Order Proximity Preserved Network Embedding. 2778–2786.

[86] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-order proximity preserved network embedding. In *SIGKDD*. 2778–2786.

[87] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. 2018. Spectral collaborative filtering. In *RecSys*. 311–319.

[88] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. 2017. Scalable graph embedding for asymmetric proximity. In *AAAI*. 2942–2948.

[89] Dingyuan Zhu, Peng Cui, Daixin Wang, and Wenwu Zhu. 2018. Deep Variational Network Embedding in Wasserstein Space. In *KDD*. 2827–2836.