FREDE: Linear-Space Anytime Graph Embeddings

Anton Tsitsulin
University of Bonn

Marina Munkhoeva
Skoltech

Panagiotis Karras
Aarhus University

Ivan Oseledets
Skoltech

Davide Mottin
Aarhus University

Emmanuel Müller
University of Bonn

ABSTRACT

Low-dimensional representations, or embeddings, of a graph’s nodes facilitate data mining tasks. Known embedding methods explicitly or implicitly rely on a similarity measure among nodes. As the similarity matrix is quadratic, a tradeoff between space complexity and embedding quality arises; past research initially opted for heuristics and linear-transform factorizations, which allow for linear space but compromise on quality; recent research has proposed a quadratic-space solution as a viable option too.

In this paper we observe that embedding methods effectively aim to preserve the covariance among the rows of a similarity matrix, and raise the question: is there a method that combines (i) linear space complexity, (ii) a nonlinear transform as its basis, and (iii) nontrivial quality guarantees? We answer this question in the affirmative, with FREDE (FREquent Directions Embedding), a sketching-based method that iteratively improves on quality while processing rows of the similarity matrix individually; thereby, it provides, at any iteration, column-covariance approximation guarantees that are, in due course, almost indistinguishable from those of the optimal row-covariance approximation by SVD. Our experimental evaluation on variably sized networks shows that FREDE performs as well as SVD and competitively against current state-of-the-art methods in diverse data mining tasks, even when it derives an embedding based on only 10% of node similarities.

1 INTRODUCTION

Graph embeddings [10, 17, 19, 28, 29, 35] empower data practitioners with a multi-purpose tool for performing a plethora of tasks, such as community detection, link prediction, and node classification. A graph embedding is a low-dimensional representation of graph’s nodes that captures the graph structure. Embeddings are popular because they eschew the burden of crafting separate features for each task. Neural graph embeddings [10, 19, 27], computed by unsupervised representation learning over nonlinear transformations, outperform their linear counterparts [17, 35] in quality.

NetMF [21] established a connection between neural graph embeddings and the factorization of a matrix of nonlinear pairwise similarities among nodes, under certain conditions on the algorithm parameters. NetMF performs Singular Value Decomposition (SVD) on a dense similarity matrix. Contrary to neural methods such as DeepWalk and node2vec, NetMF achieves the global optimum of its objective function by virtue of the properties of SVD.

Yet the optimality of NetMF comes at the price of scalability, as it needs to precompute the similarity matrix and store it in memory at quadratic cost in the number of nodes; such a method cannot be

Figure 1: FREDE produces an embedding at any time; it outperforms SVD of the full similarity matrix in 3% of the latter’s runtime, time on logarithmic x-axis. (PPI data).
We discuss the two major sources of background on which our walks; and VERSE \[27\] generalized it to a method that preserves any work builds: graph embeddings and matrix sketching.

### Table 1: Comparison of works in terms of fullfilled (✔) and missing (✘) desiderata; complexities in terms of # nodes \(n\), dimensionality \(d\), context size \(T\), and # negative samples \(b\).

| method          | Nonlinear | Closed-form | Error-bounded | Versatile | Solution | Frugal | Anytime | Mergeable | Complexity                |
|-----------------|-----------|-------------|---------------|-----------|----------|--------|---------|-----------|---------------------------|
| DeepWalk [19]   | ✔         | ✘           | ✘             | ✘         | ✔        | ✘      | ✘       | ✘         | \(O(nm)\) \(O(dn log n)\) |
| node2vec [10]   | ✔         | ✘           | ✘             | ✘         | ✔        | ✘      | ✘       | ✘         | \(O(n^2)\) \(O(dnb)\)     |
| LINE [24]       | ✔         | ✘           | ✘             | ✘         | ✔        | ✘      | ✘       | ✘         | \(O(nm)\) \(O(dn^2)\)      |
| HOPE [17]       | ✘         | ✔           | ✔             | ✔         | ✘        | ✘      | ✘       | ✘         | \(O(nm)\) \(O(dn^2)\)      |
| AROPE [35]      | ✘         | ✔           | ✘             | ✔         | ✘        | ✘      | ✘       | ✘         | \(O(nm)\) \(O(dn^2)\)      |
| VERSE [27]      | ✔         | ✘           | ✘             | ✘         | ✔        | ✘      | ✘       | ✘         | \(O(nm)\) \(O(dn^2)\)      |
| NetMF [21]      | ✔         | ✔           | ✔             | ✔         | ✘        | ✘      | ✘       | ✘         | \(O(nm)\) \(O(dn^2)\)      |
| NetSMF [20]     | ✔         | ✔           | ✔             | ✔         | ✘        | ✘      | ✘       | ✘         | \(O(nm)\) \(O(dn^2)\)      |
| FREDE (ours)    | ✔         | ✔           | ✔             | ✔         | ✔        | ✔      | ✔       | ✔         | \(O(nm)\) \(O(dnm)\)       |

2. **RELATED WORK**

We discuss the two major sources of background on which our work builds: graph embeddings and matrix sketching.

#### 2.1 Graph Embeddings

**Neural graph embeddings.** Advances in natural language processing [11, 15] provided scalable methods that derive vector representations of words. DeepWalk [19] imported such methods to graphs by materializing a corpus of random walks, treating nodes encountered on the walk as words in a text. LINE [24] extended DeepWalk by exploiting graph edges rather than random walks; node2vec [10] boosted it with a customizable generation of random walks; and VERSE [27] generalized it to a method that preserves any similarity measure among nodes, with Personalized PageRank [18] as the default option. Such neural graph embeddings reach scalability through stochastic gradient descent and sampling; they provide no closed-form solution, and hence do not offer any comprehensible quality guarantees either.

**Matrix factorization embeddings.** In another vein, matrix factorization relies on the explicit decomposition of similarity matrices among nodes. GraRep [6] factorizes, by Singular Value Decomposition (SVD), the concatenation of dense log-transformed DeepWalk transition probability matrices over different numbers of steps. GraRep is neither scalable, nor interpretable, as the concatenation provides no guarantees on the representation. HOPE [17] overcomes the scalability drawback by applying a generalized form of SVD on special similarity matrices in the form \(A^{1/2}\); HOPE achieves optimality due to the SVD guarantees of the Eckart–Young–Mirsky theorem, but its overall performance is hindered by its linear nature [27]. AROPE [35] applies spectral filtering on symmetric similarity matrices; thereby it forfeits SVD-based guarantees.

**Connecting neural and factorization worlds.** Recently, NetMF [21] extended an analysis of word embeddings [11] to suggest a connection between matrix factorization and neural embeddings: under certain probability independence assumptions, DeepWalk, LINE, and node2vec implicitly apply SVD on dense log-transformed similarity matrices. NetMF proposes novel closed-form solutions to compute such similarity matrices directly with optimal error guarantees.

However, it yields a prohibitive complexity that hinders its application to large graphs with more than 100,000 nodes. NetSMF [20] sought to improve NetMF’s scalability by sparsifying the similarity matrix while forfeiting optimality; however, the sparsified matrix has \(O(Tm log n)\) nonzeros, yielding quadratic growth in the worst case.

**Synopsis.** We focus on embeddings for graphs with only nodes and edges and with no additional information. We start out by providing, in Table 1, a comparison of related work in terms of desirable characteristics of the embedding solution and of its computation, as well as time and space requirements.

- **nonlinear:** applying nonlinear transformations; most embedding methods are nonlinear except for HOPE [17] and AROPE [35].
- **closed-form:** providing an explicit formula for the solution; NetMF [21] and NetSMF [20] are the only methods that are both nonlinear and closed-form.
- **error-bounded:** returning a solution with optimal or nontrivial error guarantees; closed-form methods are error-bounded, except for AROPE [35] and NetSMF [20] that abandon such guarantees for the sake of scalability.
- **versatile:** accommodating any similarity measure among graph nodes; DeepWalk [27], LINE [24], and node2vec [10] lack such versatility.
- **frugal** (space-efficient): having worst-case space complexity subquadratic in the number of nodes; node2vec [10], NetMF [21] and NetSMF [20] fail in that respect.
- **anytime:** allowing the computation of a partial embedding whose quality improves as more nodes are processed. Section 4.3 shows that FREDE enjoys this property.
- **mergeable:** conducive to combining embeddings on two node subsets while retaining theoretical guarantees; this property allows for distributed computation [2] with guarantees.

#### 2.2 Matrix sketching

**Matrix sketching** [4, 5, 7, 14] finds a lower-rank matrix that preserves the covariance \(M^TM\) of a matrix \(M \in \mathbb{R}^{n \times f}\) with \(f\) features and \(s\) elements. Such works operate in streaming fashion and guarantee the quality on the covariance matrix when rows arrive one after another.
The current state of the art in matrix sketching, Frequent Directions (FD) [14], extends the Misra-Gries algorithm [16] from frequent items to matrices and outperforms other methods [4, 5, 7] in quality. We introduce this streaming matrix sketching know-how to graph embeddings to build an anytime embedding algorithm that inherits the same error guarantees. Recent work [34] adapted an older matrix sketching algorithm [4, 30] to produce graph embeddings, albeit without inheriting its error guarantees; in our experiments, we use a refined variant of [34], with error guarantees, as a baseline.

3 PRELIMINARIES AND PROBLEM SETTING

Here, we show that graph embedding methods implicitly minimize the approximation error in relation to a similarity matrix among graph nodes. Arguably, this interpretation is more economic than explaining embeddings in terms of minimizing reconstruction error [21]. Then, in Section 4, we describe FREDE, a novel, anytime algorithm that offers linear space complexity and error guarantees through covariance sketching.

3.1 Problem setting and notation

A graph is a pair \( G = (V, E) \) with \( n \) vertices \( V = \{v_1, \ldots, v_n\} \), \( |V| = n \), and edges \( E \subseteq V \times V \), \( |E| = m \). A graph is represented by an adjacency matrix \( A \) for which \( A_{ij} = 1 \) if \((i, j) \in E\) is an edge between node \( i \) and node \( j \), otherwise \( A_{ij} = 0 \). The matrix \( D \) is the diagonal matrix with the degree of node \( i \) as entry \( D_{ii} \), i.e., \( D_{ii} = \sum j=1 \, A_{ij} \).

The normalized adjacency matrix is the matrix \( P = D^{-1}A \) that represents the transition probability from one node to any of its neighbors. We represent arbitrary interactions among nodes with a similarity matrix \( S \in \mathbb{R}^{n \times n} \), as done in previous work [17, 27, 35]. The row \( i \) of a matrix \( A \) is denoted as \( A_i \).

We seek to find a \( d \)-dimensional embedding, represented as a \( n \times d \) matrix \( W \) that provably retains most of the information in \( S \). We formalize this objective in Section 4.

3.2 Graph embeddings as matrix factorization

We now cast the problem of learning graph embeddings as matrix factorization. We first introduce the problem of factorization by minimization of the approximation error to pave the way on the study of the covariance error and its properties.

One way to preserve the similarities contained in the matrix \( S \) is to find an approximate matrix \( \hat{S} \) that minimizes the reconstruction error [17, 21, 35].

Definition 1 (Reconstruction error). The reconstruction error between \( S \) and \( \hat{S} \) is the Frobenius norm of the difference among the two matrices \( S \) and \( \hat{S} \), i.e., \( \| S - \hat{S} \|^2_F = \sum_{i=1}^n \sum_{j=1}^n (S_{ij} - \hat{S}_{ij})^2 \).

In other words, the reconstruction error acts element-wise and discards row dependencies.

In the case of \( S \) symmetric, there exists an eigendecomposition \( S = U \Sigma V^T \). The optimal rank-\( k \) approximation \( |S|_k = WW^T \) of \( S \) minimizing the reconstruction error is \( W = U_k \sqrt{\Sigma_k} \), obtained by the product of the first \( k \) eigenvectors \( U_k \) and a diagonal matrix of square roots of the first \( k \) eigenvalues \( \Lambda_k \).

The singular value decomposition \( S = U \Sigma V^T \), where \( U_k \) and \( V_k \) is a shorthand notation for the first \( k \) columns of \( U \) and \( V \), respectively.

The analysis in NetMF [21] shows that the Deepwalk [19] objective is equivalent to SVD on the dense similarity matrix

\[
S = \log \left( m \sqrt{T} (\sum_{r=1}^{T} P^r) D^{-1} \right),
\]

where \( T \) is the window size for the random-walk and \( b \) is the number of negative samples. The \( d \)-dimensional DeepWalk embedding is the result of multiplying the \( d \) left singular vectors \( U_d \) by the square root of the first \( d \) singular values \( \Sigma_d \), i.e., \( U_d \sqrt{\Sigma_d} \).

The chief drawback of this approach is that it requires \( O(n^2) \) space to store the dense similarity matrix \( S \). An attempt to ameliorate this drawback by sparsifying the matrix causes a loss of optimality guarantees with a deleterious effect on performance for effectual sparsity levels [20].

3.3 Matrix sketching

As the linear baseline described above is detrimental to quality, we turn our attention to an alternative approach, namely matrix sketching. Matrix sketching aims at a low-dimensional sketch \( W \in \mathbb{R}^{d \times k} \) of a matrix \( M \in \mathbb{R}^{n \times t} \) that retains most of the information in \( M \) without striving for matrix reconstruction; in our case, the matrix \( M \) corresponds to the similarity matrix \( S \). The two most popular objectives for such sketches are the minimization of covariance error and projection error.

Definition 2 (Covariance error). The covariance error is the normalized difference between the covariance matrices

\[
\text{ce}_k(M, W) = \frac{\| M^T M - W^T W \|_F}{\| M - [M]_k \|_F^2} \leq \frac{\| M^T M - W^T W \|_F}{\| M \|_F^2} = \text{ce}(M, W).
\]

Intuitively, the covariance error accounts for variance loss in each dimension. The correct \( k \) for the best rank \( k \) approximation \( [M]_k \) is not known and often requires to perform grid-search. Hence, we use the upper bound \( \text{ce}(M, W) \) as a replacement of the covariance error \( \text{ce}_k(M, W) \).

On the other hand, the projection error shows how accurate the subspace of \( W \) is in approximating true rank \( k \) subspace:

Definition 3 (Projection error). The projection error is the deviation between the matrix \( M \) and the projection \( \pi_k(M) \) of \( M \) onto the top \( k \) right singular vectors \( V_k \) of sketch \( W \)

\[
\text{pe}_k(M, W) = \frac{\| M - \pi_k(M) \|_F^2}{\| M - [M]_k \|_F^2}.
\]

The minimization of reconstruction error by SVD also yields optimal covariance and projection errors. Specifically, given the decomposition \( [M]_k = U_k \Sigma_k V_k^T \), if we set \( W = \Sigma_k^{1/2} V_k \), then \( \text{pe}_k(M, W) = 1 \) and the covariance error depends on the singular value decay of \( M \). However, SVD is often computationally prohibitive; on the other hand, sketching algorithms compute a \( W \) with error guarantees on ce and pe.

Many sketching techniques allow row-wise processing of the matrix \( M \) with quality guarantees. This property is valuable, as it allows anytime updates (Section 4.3). Another desirable property is...
mergeability, i.e., the possibility to merge multiple embeddings on the same data and retain the guarantees.

**Definition 4. Mergeability.** A sketching algorithm sketch is mergeable if there exists an algorithm merge that, applied on the \(d \times t\) sketches, \(W_1 = \text{sketch}(M_1)\) and \(W_2 = \text{sketch}(M_2)\), of two \(\frac{d}{2} \times t\) matrices, \(M_1, M_2\), with \(\text{ce}(M_1, W_1) \leq \epsilon\) and \(\text{ce}(M_2, W_2) \leq \epsilon\), produces a \(d \times t\) sketch \(W\) of the concatenated matrix \(M = [M_1; M_2]\). \(W = \text{merge}(W_1, W_2) = \text{sketch}(M)\), that preserves the covariance error bound \(\epsilon\), i.e., \(\text{ce}(M, W) \leq \epsilon\).

We now discuss some representative sketching algorithms in terms of their error bounds, efficiency, and mergeability.

**Hashing.** We construct two universal hash functions, one 2-universal function \(h : [s] \to [d]\) and one 4-universal function \(g : [s] \to \{-1, +1\}\). The sketch matrix \(M\) initially contains zeros in all entries. Then, each row \(M_i\) is added to \(h(i)\)-th row of the sketch matrix with the sign determined by \(g(i)\): \(W_{h(i)} = g(i) \cdot M_i\) meaning its complexity is linear in the matrix size \(O(s)\).

By setting \(d = O(t^2/\epsilon^2)\), hashing achieves \(\text{ce} \leq \epsilon\) [31]. This sketch is trivially mergeable: \(\text{merge}(W_1, W_2) = W_1 + W_2\).

**Random Projections.** Random projections are a foundational tool for approximate data analysis [31]. Boutsidis et al. [4] propose a row-streaming algorithm for matrix sketching that randomly combines rows of the input matrix. In matrix form, \(M = R\), where the elements \(r_{ij}\) of the \(d \times s\) matrix \(R\) are uniformly from \([-1/\sqrt{d}, 1/\sqrt{d}]\). The row-streaming algorithm works as follows: for each row \(M_i\) sample a random vector \(r_i \in \mathbb{R}^d\) with entries from the set \([-1/\sqrt{d}, 1/\sqrt{d}]\) and update \(W = W + r_i M_i^\top\).

This sketch achieves \(\text{ce} \leq \epsilon\) with \(d = O(t/\epsilon^2)\) which shaves a factor \(s\) from the theoretical guarantee of hashing-based sketches. However, the practical performance of the random projection algorithms greatly exceeds the theoretical guarantee [13]. This sketch is also mergeable as \(\text{merge}(W_1, W_2) = W_1 + W_2\).

**Sampling.** Selecting a small column subset of the entire matrix is known as the Column Subset Selection Problem (CSSP) [5]. In the row-update model the solution can be found by sampling scaled rows \(M_i/\sqrt{p_i}\) from the input matrix with probability \(p_i\) proportional to their squared \(L_2\) norm, i.e. \(p_i = \|M_i\|^2/\|M\|^2\). While the norm \(\|M\|^2\) is usually unknown in advance, the method can work with \(d\) reservoir samplers, where \(d\) is the sketch size.

This sketch achieves \(\text{ce} \leq \epsilon\) with setting \(d = O(t/\epsilon^2)\), again providing better guarantee than hashing-based sketch, however, the cost of maintaining reservoir samples is non-negligible. The sketch is mergeable if we use distributed reservoir sampling.

**Frequent Directions.** This algorithm can be seen as an extension of the item frequency approximation in streams. Frequent Directions (FD) sketches a matrix by iteratively filling the sketch with the incoming rows, performing SVD on the sketch when the sketch cannot add more rows, and shrinking the accumulated vectors with a low-rank SVD approximation.

The complexity of FD is \(O(dts)\), due to \(t/d\) iterations of computing the \(O(d^2t)\) SVD decomposition of a \(2d \times t\) matrix \(W\) with \(d \ll t\). This sketch achieves \(\text{ce} \leq \epsilon\) when \(d = O(t/\epsilon)\) and is mergeable since \(\text{merge}(W_1, W_2) = \text{FD}(\text{concatenate}(W_1, W_2))\), i.e. we obtain the merge result by invoking FD on the concatenation of the sketches.

The table below summarizes the embedding dimension \(d\) required to attain an error bound \(\text{ce} \leq \epsilon\) with \(0 < \epsilon \leq 1\) for the different sketching algorithms.

| Algorithm | Hashing | RP | Sampling | FD |
|-----------|---------|----|----------|----|
| Dimension \(d\) | \(O(t^2/\epsilon^2)\) | \(O(t/\epsilon^2)\) | \(O(t/\epsilon^2)\) | \(O(t/\epsilon)\) |

Table 2: Embedding dimension needed to achieve \(\text{ce} \leq \epsilon\) for different sketching algorithms

**4 ANYTIME GRAPH EMBEDDINGS**

Here, we first show a relationship between embeddings methods based on SVD and the preservation of the covariance of the similarity matrix. Inspired by this relationship, we propose a method that uses a state-of-the-art matrix sketching algorithm, which specifically aims to preserve covariance, to incrementally construct a covariance-preserving graph embedding in an anytime fashion. This sketching algorithm can work by row updates; thus, if the input similarity matrix \(S\) can be partially materialized, our method can compute it by row updates, gaining space efficiency. It is tempting to apply our method for a linear-space, anytime version of NetMF. Yet, the matrix form of DeepWalk, used in NetMF [21] and discussed in Section 3.2 (in Equation 1), cannot be partially materialized, hence we cannot apply our method on it. We may use any partially materializable matrix instead. As a default case, we propose a matrix \(S\) based on the Personalized PageRank (PPR) measure.

**4.1 Sketching in place of SVD**

Embedding methods based on SVD, i.e., HOPE [17] and NetMF [21], only use one of the two unitary matrices that SVD produces, \(U\) and \(V\), and discard the other. For example, NetMF returns \(W = UV^\top\) as the network embedding, where \(\Sigma\) is truncated to \(d\) singular values. Judging solely by the products of SVD they use, we conclude that these methods do not effectively aim to reconstruct the original matrix \(S\). The most relevant objective we can achieve using the SVD products of \(U\) and \(\Sigma\) is to reconstruct the row-covariance matrix \(SS^\top = U\Sigma^2U^\top\); indeed, setting \(W = U\Sigma\), without truncating \(\Sigma\), achieves this objective with zero error, as then \(WW^\top = U\Sigma^2U^\top\).

On the other hand, sketching algorithms aim to reconstruct the column-covariance matrix \(S^\top S = V\Sigma^2V^\top\), i.e., minimize the error \(\|S^\top S - W^\top W\|^2\); this error has been used as the Pairwise Inner Product (PIP) measure of similarity among word embeddings [32]. We discern a resemblance between the objective of sketching algorithms, on the one hand, and the way embedding methods use SVD, on the other hand. We venture to exploit this resemblance by applying sketching in place of SVD for embedding purposes.

**4.2 A row-wise computable similarity matrix**

HOPE [17] was the first embedding method based on any chosen higher-order similarity matrix; in its default version, it uses Katz similarity; however, it requires the entire matrix as input,
and applies a linear, rather than nonlinear, dimensionality reduction method on it via a revised form of SVD. VERSE [27] was the first similarity-based embedding method that does not require the entire matrix as input, as it allows for efficient row-wise computation; in its default version, it use Personalized PageRank (PPR) as a higher-order similarity measure and outperforms other methods in downstream tasks [27].

**Definition 5.** Given a starting node distribution $s$, damping factor $\alpha$, and the transition probability matrix $P$, the Personalized PageRank vector $PPR_i$ is defined by the recursive equation:

$$PPR_i = \alpha s + (1 - \alpha)PPR_i^TP$$

(2)

One way to compute $PPR_i$ is by means of power iteration: starting with an uniform initial guess and solving Equation (2) iteratively. Instead, we leverage the fact that the probability of a random walk with restart converges to $PPR$ with the bias parameter $b$ equal to 1 as in NetMF [21]. We treat the rows of the PPR matrix as a column basis for our embedding $(n \times d)$, and apply the Frequent Directions row-update sketching algorithm (Section 3.3) to obtain FREDE, a row-streaming embedding algorithm that computes a $d \times n$ sketch $W$ of the matrix $S$. We process rows of the PPR matrix and update the sketch accordingly.

4.3 FREDE algorithm

As the matrix $XX^T$ has equal row and column ranks, we rewrite the decomposition commutatively, as $\log(PPR) + \log n - \log b = XX^T$. We keep the bias parameter $b$ equal to 1 as in NetMF [21]. We treat the rows of the PPR matrix as a column basis for our embedding $(n \times d)$, and apply the Frequent Directions row-update sketching algorithm (Section 3.3) to obtain FREDE, a row-streaming embedding algorithm that computes a $d \times n$ sketch $W$ of the matrix $S$. We process rows of the PPR matrix and update the sketch accordingly.

**Algorithm 1 FREDE algorithm**

1. function FREDE$(G, n, d)$
2. \hspace{1cm} $W \leftarrow$ zeros$(2d, n)$ \hspace{1cm} $\triangleright$ all zeros matrix $W \in \mathbb{R}^{2d \times n}$
3. \hspace{1cm} $\hat{\Sigma} \leftarrow I(2d)$ \hspace{1cm} $\triangleright$ identity matrix $\hat{\Sigma} \in \mathbb{R}^{2d \times 2d}$
4. \hspace{1cm} for $v \in V$ do
5. \hspace{2cm} $x \leftarrow$ PersonalizedPageRank$(v)$
6. \hspace{2cm} $x \leftarrow \log x + \log n$ \hspace{1cm} $\triangleright$ VERSE similarity row
7. \hspace{2cm} insert $x$ into the last zero valued row of $W$
8. for $W$ has no zero valued rows then
9. \hspace{3cm} $U, \Sigma, V^T \leftarrow \text{SVD}(\hat{\Sigma}W)$, $\sigma \leftarrow \Sigma_{d,d}$
10. \hspace{3cm} $\hat{\Sigma}_{d,d} \leftarrow \sqrt{\max(\Sigma_{d,d}^2 - \sigma^2I_d, 0)} \times$ set $d$th row of $\hat{\Sigma}$ to 0
11. \hspace{3cm} $\hat{\Sigma}_d \leftarrow I_d$ \hspace{1cm} $\triangleright$ set last $d$ entries of $\hat{\Sigma}$ to 1
12. \hspace{3cm} $W_d \leftarrow V_d^T \hat{\Sigma}_d W_d \leftarrow 0_{d \times n}$ \hspace{1cm} $\triangleright$ last zero $d$ rows of $W$
13. return $\hat{\Sigma}, W[1:d, :]$
14. function GetEmbedding$(k \leq d)$ \hspace{1cm} $\triangleright$ Anytime
15. \hspace{1cm} return $\hat{\Sigma}W^T[1:k, :]$ \hspace{1cm} $\triangleright$ first $k$ rows

Algorithm 1 presents the details of FREDE, which computes rows of the PPR matrix by sampling, as in [27], applies Frequent Directions for sketching, and returns embeddings with guarantees at any time. Upon a request for output, we perform SVD on $W$. We lower the $O(d^2n)$ time that it would take to perform SVD for output to $O(dn^2)$ by keeping track of singular values alongside the sketch, so as to avoid recalculations at the time of output. The time to process all nodes is $O(dn^2)$. In line with previous works [17, 21], we use the square root of singular values; thus, at the time of output, we multiply by $\sqrt{\Sigma}$ where a sketching algorithm would multiply by $\Sigma$. Raising the singular values to a power in the interval $[0, 0.5]$ has been shown beneficial for downstream tasks [12, 23].

FREDE produces embeddings with error guarantees after it has materialized only a part of the similarity matrix, since covariance error bounds apply for a subset of the processed rows. Formally, we define an anytime guarantee as follows:

**Definition 6.** Embedding with Anytime Guarantees. An embedding algorithm provides anytime guarantees if a covariance error bound holds, for an arbitrary subset of nodes, even after it has processed only that subset of nodes.
As Section 3.3 outlined, anytime guarantees hold for all sketching methods and are inherited by any sketch-based node embedding. FREDE obtains more accurate covariance estimation than other sketch-based embeddings thanks to the superior error guarantees of Frequent Directions (Table 2). In particular, it achieves $ce \leq e$ on the submatrix $S_{i,j}$ built from any size-$s$ subset of processed rows (i.e., nodes) when $d=O(n/e)$ [9], independently of $s$ (Table 2). As we show in Section 5.5, FREDE outperforms other sketch-based node embeddings in the task of anytime node classification.

5 EXPERIMENTS

We evaluate FREDE against state-of-the-art graph embedding algorithms, three sketching baselines, NetMF [21] and SVD exact matrix factorization. For reproducibility purposes, we publish the source code as well as the data used for the experiments in the GitHub repository¹. We run all experiments on commodity servers with $2 \times 20$ core Intel E5-2698 v4 CPU and 384Gb RAM, with a 64Gb memory constraint for each method.

Competing methods. We exclude from the comparison methods based on linear transformations, i.e., HOPE [17] and AROPE [35], as previous work [27] has established they are outperformed by nonlinear methods. We evaluate FREDE against state-of-the-art graph embedding methods based on nonlinear transformations:

- **DeepWalk** [19]: This approach learns an embedding by sampling fixed-length random walks from each node and applying word2vec-based learning on those walks. DeepWalk remains highly competitive when used with its default parameters [27], i.e., walk length $t = 80$, number of walks per node $y = 80$, and window size $T = 10$; we use exactly these time-tested parameters.
- **VERSE** [27]: This approach learns similarity measures via sampling by training a single-layer neural network. We use personalized PageRank as similarity measure with the default parameters described in the paper, i.e., $\alpha = 0.85$ and $nsamples = 10^6$.
- **NetMF** [21]: This approach computes the DeepWalk matrix with a closed-form solution and performs SVD on that matrix. To word2vec-based learning on those walks.

Embedding baselines. We use the full SVD decomposition of the PPR matrix with the same parameters as in FREDE. Since this method is not scalable, we only evaluate it on the three smallest datasets. We additionally compare with Hashing, Random Projections and Sampling, the three high-performance baseline sketching methods described in Section 3.3. For each of those, we compute the sketch and filter singular value as in FREDE. Our Random Projections baseline is a refined variant of [34], substituting a crude higher-order matrix approximation with the row-update random projections sketching algorithm applied on a PPR matrix.

Parameter settings. We set default embedding dimension $d = 128$ for all the methods unless indicated otherwise [10, 19], and use $\alpha = 0.85$ for personalized PageRank for the default value [18, 27].

We use Intel MKL library to perform SVD using the function gesdd. For classification we use LIBLINEAR [8]. We repeat each experiment 10 times and evaluate each embedding 10 times in order to reduce the noise introduced by the embedding and classification processes.

Datasets. We assess our methods on 8 real datasets; the data characteristics are described in Table 3.

| dataset | $|V|$ | $|E|$ | $|L|$ | Avg. deg. | Density |
|---------|------|------|------|---------|---------|
| PPI     | 4k   | 77k  | 50   | 19.9    | 5.1 $\times$ 10^{-3} |
| POS     | 5k   | 185k | 40   | 38.7    | 8.1 $\times$ 10^{-3} |
| BlogCatalog | 10k | 334k | 39   | 64.8    | 6.3 $\times$ 10^{-3} |
| CoCite  | 44k  | 195k | 15   | 8.86    | 2.0 $\times$ 10^{-4} |
| CoAuthor| 52k  | 178k | —    | 6.94    | 1.3 $\times$ 10^{-4} |
| VK      | 79k  | 2.7M | —    | 34.1    | 8.7 $\times$ 10^{-4} |
| Flickr  | 80k  | 12M  | 195  | 146.55  | 1.8 $\times$ 10^{-3} |
| YouTube | 1.1M | 3M   | 47   | 5.25    | 9.2 $\times$ 10^{-6} |

¹ https://github.com/WSDM-FREDE/FREDE

5.1 Sketching quality

The optimal rank-$k$ covariance approximation can be obtained via SVD on the full similarity matrix $SS^\top = V_d S_d^2 V_d^\top$. The other sketching algorithms (Section 3.3), i.e., Hashing, Random Projections and Sampling try to produce approximations of the covariance.

We numerically compute the covariance using the definition in Section 4.1 and rank 10 projection errors ($pe_{10}$) on the smallest dataset, PPI. Figure 2 reports the covariance error ($ce$) and the
projection error ($pe_{10}$) for each method as a function of the dimension $d$. FREDE outperforms other sketching baselines by at least 2 orders of magnitude in covariance error and approaches the optimal SVD solution as the dimensionality $d$ increases. Remarkably, FREDE achieves error almost indistinguishable from the optimal and clearly outperforms the other sketchers.

### 5.2 Node classification

We now turn our attention on the quality of FREDE on node classification task. In a partially labelled graph, node classification aims at predicting the correct labels for the unlabelled nodes. We report results, whether possible, for all the methods on PPI, POS, BlogCatalog, CoCit, Flickr, and YouTube graphs.

#### Table 4: Micro-F1 classification, PPI data.

| method  | 10%  | 30%  | 50%  | 70%  | 90%  |
|---------|------|------|------|------|------|
| DeepWalk | 16.33| 19.74| 21.34| 22.39| 23.38|
| NetMF    | 18.58| 22.01| 23.87| 24.65| 25.30|
| VERSE   | 16.45| 19.89| 21.64| 23.08| 23.84|
| FREDE   | 19.56| 23.11| 24.38| 25.11| 25.52|
| SVD     | 18.31| 22.12| 23.66| 25.03| 25.78|
| Rand. Proj. | 16.80| 19.99| 21.45| 22.38| 23.14|
| Sampling | 16.25| 19.55| 20.93| 21.85| 22.68|
| Hashing | 16.73| 19.97| 21.51| 22.43| 23.44|

#### Table 5: Micro-F1 classification, BlogCatalog data.

| method  | 1%  | 3%  | 5%  | 7%  | 9%  |
|---------|-----|-----|-----|-----|-----|
| DeepWalk | 36.22| 39.84| 41.22| 42.06| 42.53|
| NetMF    | 36.62| 39.80| 41.05| 41.70| 42.17|
| VERSE   | 35.82| 40.06| 41.63| 42.63| 43.14|
| FREDE   | 35.69| 38.88| 39.98| 40.54| 40.75|
| SVD     | 37.60| 40.99| 42.10| 42.66| 43.47|
| Rand. Proj. | 30.82| 34.43| 35.81| 36.52| 37.16|
| Sampling | 29.44| 32.32| 33.41| 34.04| 34.29|
| Hashing | 30.81| 34.36| 35.82| 36.65| 37.28|

#### Table 6: Micro-F1 classification, Flickr data.

| method  | 1%  | 3%  | 5%  | 7%  | 9%  |
|---------|-----|-----|-----|-----|-----|
| DeepWalk | 32.39| 36.02| 37.41| 38.15| 38.70|
| VERSE    | 30.08| 34.22| 36.06| 37.11| 37.83|
| FREDE  | 30.90| 32.98| 33.86| 34.48| 34.88|
| Rand. Proj. | 28.92| 32.21| 33.82| 34.76| 35.49|
| Sampling | 28.46| 30.97| 32.08| 32.75| 33.24|
| Hashing | 29.07| 32.23| 33.77| 34.75| 35.48|

#### Figure 2: Covariance (left) and projection (right) errors with varying embeddings dimensionality $d$. 

![Covariance Error](image1)

![Projection Error](image2)

We evaluate the accuracy in terms of Micro-F1 and Macro-F1 measures as it is common in the literature [19, 24]. Due to space restriction, we only report Micro-F1 results since the behavior of
Macro-F1 results is similar across datasets. For each dataset, we repeat the experiment 10 times and report the average. Tables 4, 5, 6, 7, 8, 9 report the results on different datasets; SVD is featured in those cases where it could run within 64Gb.

Our results confirm the superiority of our similarity matrix sketching over NetMF factorization, across the tested datasets. Surprisingly, on PPI and POS, FREDE outperforms its theoretically grounded competitor, SVD. Consistently across all datasets, FREDE is better than its sketching counterparts.

5.3 PPR approximation

Here, we evaluate the effectiveness of approximating PPR using random walks. Figure 3 shows the downstream performance of the sketching algorithms as the number of random walks for PPR approximation grows. FREDE performs consistently better than sketching baselines, while it reaches the exact-PPR solution in quality with $10^6$ walks. This result indicates that we can achieve performance obtained using the exact PPR values in downstream tasks even without computing such PPR values with high precision.

5.4 Link prediction

Link prediction is the task of predicting the appearance of a link between pairs of nodes in a graph. We evaluate link prediction on two datasets – CoCit and VK (Tables 11, 12). As a baseline, we train a logistic regression classifier on traditional link prediction features (node degree, number of common neighbors, Adamic-Adar index, Jaccard coefficient, and preferential attachment index). Features are constructed from embeddings according to the rules outlined in the

### Table 9: Micro-F1 classification, YouTube data.

| method      | 1%   | 3%   | 5%   | 7%   | 9%   |
|-------------|------|------|------|------|------|
| DEEPWALK    | 37.96| 40.54| 41.72| 42.60| 43.37|
| VERSE       | 38.04| 40.50| 41.72| 42.59| 43.33|
| FREDE       | 34.51| 37.37| 38.78| 39.40| 39.95|
| Rand. Proj. | 33.88| 36.10| 37.23| 37.94| 38.38|
| Sampling    | 33.97| 35.66| 36.37| 37.19| 37.71|
| Hashing     | 32.64| 35.64| 36.92| 37.46| 38.13|

### Table 10: Edge embedding strategies for link prediction task

| Operator          | Result                                                                 |
|-------------------|-------------------------------------------------------------------------|
| Average           | $(a + b)/2$                                                              |
| Concat            | $[a_1, \ldots, a_d, b_1, \ldots, b_d]$                                  |
| Hadamard          | $[a_1 \ast b_1, \ldots, a_d \ast b_d]$                                  |
| Weighted L1       | $[[a_1 - b_1], \ldots, [a_d - b_d]]$                                    |
| Weighted L2       | $[(a_1 - b_1)^2, \ldots, (a_d - b_d)^2]$                                |

### Table 10: Link prediction results on the MS coauthorship graph. Best results per method are underlined.

| method      | Average | Concat | Hadamard | L1 | L2 |
|-------------|---------|--------|----------|----|----|
| DEEPWALK    | 68.97   | 68.43  | 66.61    | 78.80 | 77.89 |
| VERSE       | 79.62   | 79.25  | 86.27    | 75.15 | 73.32 |
| FREDE       | 81.28   | 80.95  | 86.85    | 81.70 | 82.37 |
| Rand. Proj. | 80.81   | 80.54  | 86.73    | 80.79 | 81.42 |
| Sampling    | 80.98   | 80.74  | 86.45    | 79.53 | 79.51 |
| Hashing     | 80.84   | 80.48  | 86.66    | 80.59 | 81.33 |
| Baseline    | 77.53   |        |          |     |    |

### Table 12: Link prediction results on the VK social graph. Best results per method are underlined.

| method      | Average | Concat | Hadamard | L1 | L2 |
|-------------|---------|--------|----------|----|----|
| DEEPWALK    | 69.98   | 69.83  | 69.56    | 78.42 | 77.42 |
| VERSE       | 74.56   | 74.42  | 80.94    | 77.16 | 77.47 |
| FREDE       | 74.68   | 74.59  | 77.63    | 74.25 | 73.60 |
| Rand. Proj. | 74.41   | 74.27  | 77.01    | 74.33 | 74.56 |
| Sampling    | 74.38   | 74.27  | 76.82    | 72.26 | 71.95 |
| Hashing     | 74.36   | 74.27  | 76.86    | 74.30 | 74.56 |
| Baseline    | 78.84   |        |          |     |    |

5.5 Anytime classification

Here, we study the anytime operation (cf. Section 4.3) of sketch-based embeddings on node classification with 50% of nodes used for training, while processing rows of the PPR matrix in random order. Figure 4 presents our results on three datasets, juxtaposed for training, while processing rows of the PPR matrix in random order. Figure 4 shows the downstream performance of sketching algorithms on PPI data wrt. number of walks to compute PPR.
after processing only 15% of nodes. This outcome indicates the potential of sketch-based embeddings to achieve good performance without computing an entire similarity matrix. We also present runtime on the BlogCatalog data; all sketcher runtimes grow linearly in the number of processed nodes, while those of VERSE and SVD, which cannot produce anytime embeddings, stand apart.

6 CONCLUSION
We discerned that graph embedding methods aim to preserve the covariance (i.e., row dot-product) of a similarity matrix; therefrom we deduced that row-wise matrix sketching techniques are naturally suited to graph embeddings, since they preserve covariance by design. Thus, we applied a state-of-the-art sketching technique on our proposal for a matrix-factorization interpretation of a state-of-the-art embedding, to craft FREDE: a linear-space, anytime graph embedding that represents a nonlinear node-to-node similarity matrix with near-optimal covariance preservation quality. FREDE allows merging embeddings computed using different subsets of nodes into one, maintaining quality guarantees; this property allows for its incremental, anytime computation and also opens up a potential for distributed processing. Our experiments show that FREDE achieves covariance and projection errors almost as low as those of SVD and is competitive against previous embedding methods on various tasks, notwithstanding its anytime character.

REFERENCES
[1] 2016. Microsoft Academic Graph - KDD cup 2016. https://kddcup2016.azurewebsites.net/Data.
[2] Pananj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. 2013. Mergeable summaries. TODS (2013), 26:1–26:28.
[3] Bahman Bahmani, Abdul Chowdhury, and Ashish Goel. 2010. Fast incremental and personalized pagerank. (2010).
[4] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. 2011. Near Optimal Column-Based Matrix Reconstruction. FOCS (2011), 305–314.
[5] Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. 2009. An Improved Approximation Algorithm for the Column Subset Selection Problem. In SODA.
[6] Shaosheng Cao, Wei Lu, and Qiongkai Xiu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In CIKM.
[7] Kenneth L. Clarkson and David P. Woodruff. 2013. Low rank approximation and regression in input sparsity time. In STOC.
[8] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. JMLR (2008).
[9] Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P. Woodruff. 2016. Frequent Directions: Simple and Deterministic Matrix Sketching. SIAM J. Comput. 45 (2016), 1762–1792.
[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In KDD.
[11] Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In NIPS.
[12] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. TACL (2015).
[13] Ping Li, Trevor J. Hastie, and Kenneth Ward Church. 2006. Very sparse random projections. In KDD.
[14] Edo Liberty. 2013. Simple and deterministic matrix sketching. In KDD.
[15] Tomas Mikolov, Bya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffery Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In NIPS.
[16] Jayaved Misra and David Gries. 1982. Finding repeated elements. Science of Computer Programming (1982).
[17] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric Transitivity Preserving Graph Embedding. In KDD.
[18] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: bringing order to the web. (1999).
[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In KDD.
[20] Jiezong Qu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. In WWW.
[21] Jiezong Qu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In WSDM.
[22] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorry Boucher, Ashton Breitkreutz, and Mike Tyers. 2006. BioGRID: a general repository for interaction datasets. Nucleic Acids Research (2006).
[23] Karl Stratos, Michael Collins, and Daniel Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In ACL, Vol. 1. 1282–1291.
[24] Jian Tang, Meng Qu, Mingze Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In WWW.
[25] Lei Tang and Huan Liu. 2009. Scalable learning of collective behavior based on sparse social dimensions. In CIKM.
[26] Mikkel Thorup and Uri Zwick. 2005. Approximate distance oracles. JACM (2005).
[27] Anton Tuyttalin, Davide Mottin, Panagiotis Karras, and Emmanouel Muller. 2018. VERSE: Versatile Graph Embeddings from Similarity Measures. In WWW.
[28] Anton Tuyttalin, Marina Munkhoeva, Davide Mottin, Panagiotis Karras, Alex Bronstein, Ivan Oseledets, and Emmanouel Muller. 2020. The Shape of Data: Intrinsic Distance for Data Distributions. In ICLR.
[29] Anton Tuyttalin, Marina Munkhoeva, and Bryan Perozzi. 2020. Just slag when you approximate: Accurate spectral distances for web-scale graphs. In WWW.
[30] Santosh S Vempala. 2005. The random projection method. American Math. Soc.
[31] David P Woodruff. 2014. Sketching as a Tool for Numerical Linear Algebra. Theoretical Computer Science (2014).
[32] Zhi Yin and Yuanyuan Shen. 2018. On the dimensionality of word embedding. In NIPS.
[33] R. Zafarani and H. Liu. 2009. Social Computing Data Repository at ASU. http://socialcomputing.asu.edu
[34] Ziwei Zhang, Peng Cui, Haoyang Li, Xiao Wang, and Wenwu Zhu. 2018. Billion-scale Network Embedding with Iterative Random Projection. In ICDM. 787–796.
[35] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-Order Proximity Preserved Network Embedding. In KDD.
[36] Shlomo Zilberstein. 1996. Using Anytime Algorithms in Intelligent Systems. AI Magazine 17, 3 (1996), 73–83.