SimSky: An Accuracy-Aware Algorithm for Single-Source SimRank Search

Liping Yan[1] and Weiren Yu[2](✉)

[1] Nanjing University of Science and Technology, Jiangsu, China
lipingyan@njust.edu.cn
[2] The University of Warwick, Coventry CV4 7AL, UK
Weiren.Yu@warwick.ac.uk

Abstract. SimRank is a popular node-pair similarity search model based on graph topology. It has received sustained attention due to its wide range of applications in real-world scenarios. Considerable effort has been devoted to devising fast algorithms for SimRank computation through either iterative approaches or random walk based methods. In this paper, we propose an efficient accuracy-aware algorithm for computing single-source SimRank similarity. First, we devise an algorithm, ApproxDiag, to approximate the diagonal correction matrix. Next, we propose an efficient algorithm, named SimSky, which utilizes two Krylov subspaces for transforming high-dimensional single-source SimRank search into low-dimensional matrix-vector multiplications. Extensive experiments on various real datasets demonstrate the superior search quality of SimSky compared to other competitors.

Keywords: SimRank · Single-Source Similarity Search · Low-order Approximation

1 Introduction

A graph is a key structure for modeling complexity networks, in which nodes represent objects and edges represent relationships. Measuring similarity between objects is an important task in graph mining, with many real applications, e.g. link prediction [8], recommendation systems [3], web page ranking [17], and so forth. A variety of similarity measures have been proposed over the past decades, including Personalized PageRank [6], SimRank [5], RoleSim* [14], CoSimRank [10,19], CoSimHeat [20]. Among them, SimRank is considered an influential one. SimRank is based on the simple recursive concept [5] that *"two nodes are similar if their in-neighbors are similar; every node is most similar to itself"*. Let $G = (V, E)$ be a digraph with $|V|$ nodes and $|E|$ edges. We denote by $I(i) = \{j \in V | \exists (j, i) \in E\}$ the in-neighbor set of i, and $|I(i)|$ the in-degree of i. The SimRank score $s(i, j)$ between nodes i and j is defined as

$$s(i,j) = \begin{cases} 1, & i = j; \\ \frac{c}{|I(i)||I(j)|} \sum_{u \in I(i)} \sum_{v \in I(j)} s(u,v), & i \neq j; \\ 0, & |I(i)| \text{ or } |I(j)| = 0, \end{cases} \tag{1}$$

where $c \in (0,1)$ is a decay factor, typically assigned a value of 0.6 or 0.8.

SimRank Matrix Notations. Let $S^{(k)}$ be the k-th iterative SimRank matrix, where each element $[S^{(k)}]_{i,j}$ is the similarity score $s(i,j)$ at iteration k. Let A be the column-normalized adjacency matrix of a graph, and I be the identity matrix. In matrix notations, the SimRank matrix $S^{(k)}$ can be expressed as

$$\begin{aligned} S^{(k)} &= cA^T S^{(k-1)} A + D_k \\ &= \sum_{i=0}^{k} c^i (A^T)^i D_{k-i} A^i, \end{aligned} \tag{2}$$

where $S^{(0)} = D_0 = I$, and $D_k = I - (cA^T S^{(k-1)} A) \circ I$ is called the *diagonal correction matrix*. The symbol $(*)^T$ stands for matrix transpose, and $\circ$ denotes entry-wise multiplication.

Single-Source SimRank. Given a query j, single-source SimRank search returns the similarity scores between node j and each node in the graph. Mathematically, given the query vector e_j (a unit vector with only a 1 in the j-th entry, and 0 s elsewhere), the single-source SimRank vector $[S^{(k)}]_{*,j}$ at the k-th iteration can be represented as

$$[S^{(k)}]_{*,j} = S^{(k)} e_j. \tag{3}$$

Recently, many endeavors [7,9,12,13,15,18] have been invested in designing faster and more efficient algorithms for accelerating single-source SimRank computation at the expense of accuracy. The low accuracy of SimRank arises from two main barriers: (1) the challenge of dealing with the intractable diagonal correction matrix; (2) the problem of high-dimensionality in SimRank iterations.

– *Intractable Diagonal Correction Matrix.* The challenge in retrieving single-source SimRank via Eq. 2 lies in the computation of diagonal correction matrix D_k. There are studies [4,7,16] that attempt to mitigate this issue using the following equation:

$$S^{(k)} = cA^T S^{(k-1)} A + (1-c)I. \tag{4}$$

However, the similarity models represented by Eqs. 2 and 4 are different.

– *High Dimensionality.* In reality, most graphs are large and sparse, leading to the high dimensionality of the adjacency matrix. Most existing work [12,13] employs random walk-based methods through Monte Carlo sampling. While these methods excel in superior scalability on large graphs, they typically exhibit low accuracy with a certain probability. For instance, the state-of-the-art single-source SimRank algorithms (e.g. ExactSim [13] and SLING [12]) using Monte Carlo approaches can only achieve a precision level of up to 10^{-7} on diverse real datasets.

Contributions. Our main contributions to this work are summarized as follows:

- We first design an algorithm, ApproxDiag, to approximate the diagonal correction matrix D with guaranteed accuracy. To make approximation more stable, we resort to a row orthogonal matrix to characterize D (Sect. 2).
- We next propose an efficient algorithm, SimSky, which transforms high-dimensional single-source SimRank search into matrix-vector multiplications over two small Krylov subspaces, eliminating much redundancy (Sect. 3).
- We conduct extensive experiments to demonstrate the superiority of SimSky over other rivals on real datasets (Sect. 4).

2 ApproxDiag: Approximate Diagonal Correction Matrix

For any matrix $X \in \mathbb{R}^{n\times n}$, we denote by the column vectors $\overrightarrow{diag}(X)$ and $\widetilde{diag}(X)$ the exact and approximate solution of the main diagonal elements of X, respectively. Bekas et al. [1] showed that $\widetilde{diag}(X)$ can be obtained by arbitrary column vectors $w_1, w_2, \cdots, w_s \in \mathbb{R}^n$ as follows:

$$\widetilde{diag}\,(X) = [\sum_{l=1}^{s} w_l \circ (Xw_l)] \oslash [\sum_{l=1}^{s} w_l \circ w_l], \tag{5}$$

where $\oslash$ represents entry-wise division. Let $W = [w_1|w_2|\cdots|w_s]$. If WW^T is a diagonal matrix with all diagonals being nonzeros, then $\widetilde{diag}(X) = \overrightarrow{diag}(X)$.

Construct Matrix W. Bekas et al. [1] chose the matrix W as a Hadamard matrix, which takes only the entries ± 1 so that $WW^T = nI$. This type of matrix W is suitable for approximating the main diagonal elements of a band matrix. However, in practice, the graph adjacency matrix is rarely a band matrix. Therefore, we design a novel method to construct matrix $W \in \mathbb{R}^{n\times s}$ as follows:

1) $W(1:s, 1:s)$ is an identity matrix;
2) the element of $W(1+s:n, s)$ is -1 at odd positions and 1 at even positions;
3) the remaining entries in W are all 0s.

As a special case, when $s = n$, W reduces to I and $\widetilde{diag}(X) = \overrightarrow{diag}(X)$.

Subtracting the item $\sum_{i=1}^{k} c^i (A^T)^i D_{k-i} A^i$ from both sides of Eq. 2 and applying Eq. 5 yield the following equation:

$$\widetilde{diag}(D_k) = \overrightarrow{1}_n - (\sum_{l=1}^{s} w_l \circ f(A, w_l, k)) \oslash (\sum_{l=1}^{s} w_l \circ w_l), \tag{6}$$

where $f(A, w_l, k) = \sum_{i=1}^{k} c^i (A^T)^i D_{k-i} A^i w_l$.

By virtue of the idea in [18], for $k \geq 2$, we can express the vector $f(A, w_l, k)$ as follows: $\forall i = 1, 2, \cdots, k$, initialize $x_0 = w_l$,

$$x_i = Ax_{i-1}; \tag{7}$$

Algorithm 1: ApproxDiag(A, c, k)

```
Input: $A$ - column-normalised adjacency matrix, $c$ - decay factor, $k$ - number of iterations
Output: $\widehat{D}$ - contains $(k+1)$ approximate diagonal correction vectors
1  Custom matrix $W$ and set denom $= (W \circ W) \cdot \overrightarrow{1}_s$;
2  Initialise $\widehat{D} =$ zeros$(n, k+1)$ and set $\widehat{D}(:,1) = \overrightarrow{1}_n$;
3  for $j = 1$ to $k$ do
4      Initialise nume $=$ zeros$(n,1)$, $X =$ zeros$(n, j+1)$;
5      for $i = 1$ to $s$ do
6          $X(:,1) = W(:,i)$;
7          for $a = 1$ to $j$ do
8              $X(:,a+1) \leftarrow A \cdot X(:,a)$;
9          end
10         Initialise $Y =$ zeros$(n, j+1)$, set $Y(:,1) = \widehat{D}(:,1) \circ X(:,j+1)$;
11         if $j = 1$ then
12             $Y(:,j+1) \leftarrow cA^T \cdot Y(:,j)$;
13         else
14             for $b = 2$ to $j$ do
15                 $Y(:,b) \leftarrow cA^T \cdot Y(:,b-1) + \widehat{D}(:,b) \circ X(:,j+2-b)$
16             end
17             $Y(:,j+1) \leftarrow cA^T \cdot Y(:,j)$;
18         end
19         nume $\leftarrow$ nume $+ W(:,i) \circ Y(:,j+1)$;
20     end
21     $\widehat{D}(:,j+1) \leftarrow \overrightarrow{1}_n -$ nume $\oslash$ denom;
22 end
23 return $\widehat{D}$;
```

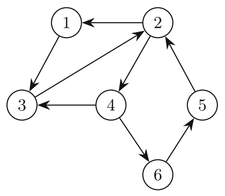
$\forall j = 1, 2, \cdots, k-1$, initialize $y_0 = \overrightarrow{diag}(D_0) \circ x_k$,

$$y_j = cA^T y_{j-1} + \overrightarrow{diag}(D_j) \circ x_{k-j}, \tag{8}$$

thus we can get $f(A, w_l, k) = cA^T y_{k-1}$ easily. Substituting $f(A, w_l, k)$ into Eq. 6, we can get our ApproxDiag algorithm.

Example 1. Given a graph and its column-normalised adjacency matrix A as shown in Fig. 1, decay factor $c = 0.8$, number of iterations $k = 2$. Take 6×2 matrix W_2, 6×3 matrix W_3, 6×4 matrix W_4, 6×5 matrix W_5 and 6×6 identity matrix W_6, where

$$W_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & 1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}, \quad W_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix}, \quad W_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad W_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}.$$



$$A = \begin{bmatrix} 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 1 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Fig. 1. A digraph with six nodes and its column-normalised adjacency matrix

According to our ApproxDiag algorithm, when W takes W_2, W_3, W_4, W_5, W_6 respectively, the corresponding matrix contains approximate diagonal correction vectors are

$$\widehat{D}_2 = \begin{bmatrix} 1 & 0.2 & 0.2 \\ 1 & 0.6 & 0.68 \\ 1 & 1 & 1 \\ 1 & 0.2 & 0.2 \\ 1 & 0.2 & 0.04 \\ 1 & 0.6 & 0.92 \end{bmatrix}, \quad \widehat{D}_3 = \begin{bmatrix} 1 & 0.2 & 0.2 \\ 1 & 0.6 & 0.52 \\ 1 & 1 & 1 \\ 1 & 0.2 & 0.2 \\ 1 & 0.2 & -0.12 \\ 1 & 0.6 & 0.92 \end{bmatrix}, \quad \widehat{D}_4 = \widehat{D}_5 = \widehat{D}_6 = \begin{bmatrix} 1 & 0.2 & 0.2 \\ 1 & 0.6 & 0.6 \\ 1 & 0.6 & 0.28 \\ 1 & 0.2 & 0.2 \\ 1 & 0.2 & 0.2 \\ 1 & 0.2 & 0.2 \end{bmatrix}.$$

□

Error Analysis. We analyze the error of $\overrightarrow{diag}(D_k)$ and $\widetilde{diag}(D_k)$. $\|\cdot\|$ represents the L_2 norm for a vector or the spectral norm for a matrix. First of all, suppose that $Z = \sum_{i=1}^{k} c^i (A^T)^i D_{k-i} A^i$, subtract Z from both sides of Eq. 2 and vectorize the main diagonal elements, $\overrightarrow{diag}(D_k) = \overrightarrow{1}_n - \overrightarrow{diag}(Z)$ can be obtained. Similarly, combine with Eq. 5, we can get that $\widetilde{diag}(D_k) = \overrightarrow{1}_n - \overrightarrow{diag}(WW^T Z)$. According to the definition of W, we know that $(WW^T - I)(1:s-1, 1:s-1) = 0$, and $(WW^T - I)_{ii} = 0$, except that all the other elements either 1 or -1. Given column-normalised adjacency matrix A, due to $c \in (0,1)$, we have spectral radius $\rho(\sqrt{c}A) < 1$. As per Theorem 5 in [2], exists a constant θ depends only on $\sqrt{c}A$ and σ, where $\rho(\sqrt{c}A) < \sigma < 1$, $\theta = \max(1, \frac{\sigma^k}{\|(\sqrt{c}A)^k\|})$, such that $\|(\sqrt{c}A)^{i-1}\| \leq \theta\sigma^{i-1}$. At the same time, it's obvious $\|\sqrt{D_{k-i}}\| \leq 1$. Finally, combine the above equalities and inequalities, the gap between $\overrightarrow{diag}(D_k)$ and $\widetilde{diag}(D_k)$ is bounded by

$$\|\overrightarrow{diag}(D_k) - \widetilde{diag}(D_k)\|_\infty \leq \theta^2 \tfrac{c(1-\sigma^{2k})}{1-\sigma^2} \max_{s \leq l \leq n} \sum_{j=s, j \neq l}^{n} \|Ae_l\| \|Ae_j\|, \tag{9}$$

where e_l (resp. e_j) is a n-dimensional unit vector with only a 1 in the l-th (resp. j-th) entry. The equal sign "=" holds when $s = n$.

Cost Overheads. We analyze the computational cost of ApproxDiag as follows. First, initializing A, W, D requires $\mathcal{O}(nd), \mathcal{O}(ns), \mathcal{O}(nk)$ memory, respectively[1]. Second, computing D (lines 3–22) take $\mathcal{O}(\sum_{j=1}^{k} s(jnd + (j-1)(n+nd)))$ time. Therefore, it requires $\mathcal{O}(n \cdot max(d,s,k))$ memory and takes $\mathcal{O}(k^2 snd)$ time.

[1] d denotes the average node degree.

3 SimSky

Yu et al. [18] demonstrated that single-source SimRank search $[S^{(k)}]_{*,j}$ can be expressed as a double loop function, we notice that it can be further rewritten as a piecewise function

$$[S^{(k)}]_{*,j} = r_{2k}, \tag{10}$$

where

$$r_l = \begin{cases} \widetilde{diag}(D_{l+1}) \circ (A^{k-1-l} e_j), & 0 \le l \le k-1; \\ \overrightarrow{diag}(D_0) \circ (A^k e_j), & l = k; \\ cA^T r_{l-1} + r_{l-1-k}, & k+1 \le l \le 2k. \end{cases} \tag{11}$$

The calculation of the last item r_{2k} can be divided into three parts to complete: (1) approximating $\widetilde{diag}(D_{l+1})$ using our ApproxDiag algorithm, (2) computing $A^{k-1-l}e_j, A^k e_j$ via the Arnoldi algorithm [11], (3) computing r_l by means of our SimSky algorithm for $k+1 \le l \le 2k$.

For $k+1 \le l \le 2k$, r_l can be expressed as

$$r_l = BR_{l-1}, \tag{12}$$

where $B = \left[cA^T \; 0 \cdots 0 \; I\right]$, $R_{l-1} = \left[r_{l-1}^T \; r_{l-2}^T \cdots r_{l-k}^T \; r_{l-1-k}^T\right]^T$. Meanwhile, we use the auxiliary equation $\left[r_{l-1}^T \; r_{l-2}^T \cdots r_{l-k}^T\right]^T = I_{kn} \cdot \left[r_{l-1}^T \; r_{l-2}^T \cdots r_{l-k}^T\right]^T$, I_{kn} is a kn-dimensional identity matrix. Concatenate the Eq. 12 and the auxiliary equation along the vertical direction, we can get the following expression

$$R_l = \begin{bmatrix} B \\ I_{kn} \;\; 0 \end{bmatrix} R_{l-1} = CR_{l-1}, \tag{13}$$

where $R_l = \left[r_l^T \; r_{l-1}^T \cdots r_{l-k+1}^T \; r_{l-k}^T\right]^T$, 0 is a $kn \times n$ null matrix, C is a sparse block matrix. Set $R_k = \left[r_k^T \; r_{k-1}^T \cdots r_1^T \; r_0^T\right]^T$, we can get that $R_{2k} = C^k R_k$, and r_{2k} is the first component of R_{2k}.

Therefore, R_k as the initial vector, C as the initial matrix, we can construct the Krylov subspace

$$\mathcal{K}_{m_2} = span\{R_k, CR_k, C^2R_k, \cdots, C^{m_2-1}R_k\},$$

then we can again use the Arnoldi algorithm [11] to compute its basis matrix and projection matrix, and calculate vector r_{2k} according to Lemma 3.1 in [11].

Example 2. Given the graph and its column-normalised adjacency matrix as shown in Fig. 1. Given the query vector e_1 is a unit vector with only a 1 in the first entry, decay factor $c = 0.8$, low-order parameters $m_1 = m_2 = 3$, number of iterations $k = 2$. For brevity, we assume that diagonal correction matrix D_{k-i} is an identity matrix. Then the process to calculate single-source SimRank search

$$[S^{(2)}]_{*,1} = c^2(A^T)^2A^2e_1 + cA^TAe_1 + e_1 \tag{14}$$

using our SimSky algorithm is as follows.

First, we convert Eq. 14 into a piecewise function

$$r_0 = Ae_1, \quad r_1 = e_1, \quad r_2 = A^2 e_1, \quad r_3 = cA^T r_2 + r_0, \quad r_4 = cA^T r_3 + r_1,$$

it's obvious that $[S^{(2)}]_{*,1} = r_4$.

Second, we construct the first Krylov subspace

$$\mathcal{K}_{m_1} = span\{r_1, r_0, r_2\} = span\{e_1, Ae_1, A^2 e_1\},$$

its column orthonormal matrix U and projection matrix Y can be generated by Arnoldi method [11], and a relationship is established

$$AU(:, 1:3) = UY.$$

As a result, according to Lemma 3.1 in [11], r_0, r_1, r_2 can be rewritten as

$$r_0 = UYe_1', \quad r_1 = U_3 e_1', \quad r_2 = UYY_3 e_1', \tag{15}$$

where e_1' is a 3-dimensional unit vector with only a 1 in the first component, matrix U_3 consists of the first three columns of matrix U, Y_3 includes the first three rows and first three columns of matrix Y.

Finally, we construct the second Krylov subspace

$$\mathcal{K}_{m_2} = span\{v, Cv, C^2 v\},$$

where block vector $v = \begin{bmatrix} r_2 \; r_1 \; r_0 \end{bmatrix}$, block matrix $C = \begin{bmatrix} B \\ I_{12} \;\; 0 \end{bmatrix}$ and $B = \begin{bmatrix} cA^T \; 0 \; I \end{bmatrix}$, I_{12} is a 12-dimensional identity matrix.

Its column orthonormal matrix Q, non-orthonormal matrix P and projection matrix H can be generated through the Arnoldi method [11] and the equality holds as follows

$$cA^T Q(:, 1:m_2) + P(:, 1:m_2) = QH.$$

Thus, r_3, r_4 can be rewritten as

$$r_3 = \|r_2\| QHe_1', \quad r_4 = \|r_2\| QHH_3 e_1',$$

where matrix H_3 includes the first three rows and first three columns of matrix H. In other words, we can transform high-dimensional single-source SimRank search r_4 into low-dimensional matrix vector multiplication $\|r_2\| QHH_3 e_1'$ to eliminate the barrier of redundant dimensionality. $\square$

Cost Overheads. We analyze SimSky's cost overheads step-by-step. At the beginning, invoking the Arnoldi algorithm takes $\mathcal{O}(m_1 nd)$ time and requires $\mathcal{O}(m_1 n)$ memory. Meanwhile, computing the scalar β takes $\mathcal{O}((k-1)m_1^2)$ time. Second, invoking the ApproxDiag algorithm takes $\mathcal{O}(k^2 snd)$ time and requires $\mathcal{O}(n \cdot max(d, s, k))$ memory. Then, initialising V, H require $\mathcal{O}(km_2 n), \mathcal{O}(m_2^2)$ memory respectively. And, setting the first column $V(:, 1)$ needs $\mathcal{O}(m_1^2 n)$ time. Finally, computing matrices V, H (lines 7–20) take $\mathcal{O}(m_2^2 kn + m_2 nd)$ time, and computing $[S_{m_1,m_2}]_{*,j}$ (line 22) takes $\mathcal{O}(m_2^2 n)$ time. Add them up, in the aggregate, it takes $\mathcal{O}((m_1 + m_2 + k^2 s)nd + (m_1^2 + km_2^2)n)$ time, requires $\mathcal{O}(n \cdot max(km_2, d, s, m_1))$ memory.

```
Algorithm 2: SimSky($A, m_1, m_2, k, c, e_j$)
Input: $A$ - column-normalised adjacency matrix, $m_1, m_2$ - low-order
       parameters, $k$ - number of iterations, $c$ - decay factor, $e_j$ - query vector
Output: $[S_{m_1,m_2}]_{*,j}$ - single-source SimRank score
1  $[U, Y, m_1] \leftarrow$ Arnoldi$(A, m_1, e_j)$;
2  Set $Y_m = Y(1:m_1, 1:m_1), \beta = \|YY_m^{k-1}e_1\| \neq 0$;
3  $D \leftarrow$ ApproxDiag$(A, c, k)$;
4  Initialise matrices $V =$ zeros$((k+1)n, m_2+1)$, $H =$ zeros$(m_2+1, m_2)$;
5  $e_1(e_2)$ is an $m_1(m_2)$-dimensional unit vector with only a 1 in the first entry;
6  Set $V(:,1) = \frac{1}{\beta}\begin{bmatrix} D(:,1) \circ (UYY_m^{k-1}e_1) \\ D(:,k+1) \circ e_j \\ \vdots \\ D(:,3) \circ (UYY_m^{k-3}e_1) \\ D(:,2) \circ (UYY_m^{k-2}e_1) \end{bmatrix}$;
7  for $i = 1$ to $m_2$ do
8      $r \leftarrow cA^TV(1:n, i) + V(kn+1:(k+1)n, i)$;
9      $t \leftarrow V(1:kn, i)$;
10     $s \leftarrow$ concatenate $r$ and $t$ along the vertical direction;
11     for $j = 1$ to $i$ do
12         $temp \leftarrow$ inner product of $r$ and $V(1:n, j)$;
13         $s \leftarrow s - temp \cdot V(:, j)$;
14         $H(j, i) \leftarrow temp$;
15     end
16     if $H(i+1, i)$ satisfies stop criterion then
17         $m_2 = i, V = V(:, 1:m_2+1), H = H(1:m_2+1, 1:m_2)$;
18     end
19     $V(:, i+1) \leftarrow \frac{s}{H(i+1,i)}$;
20 end
21 $Q = V(1:n, :), H_m = H(1:m_2, 1:m_2), P = V(kn+1:(k+1)n, :)$;
22 return $[S_{m_1,m_2}]_{*,j} = \beta QHH_m^{k-1}e_2$;
```

Error Analysis. Finally, according to different value ranges of m_1, m_2, k, we analyze the error generated by our SimSky algorithm at length. Taking into account the effects of k, m_1, m_2 on the error, we exclude the interference of diagonal correction matrix D_k on the error, that is, we suppose that W is an identity matrix. The error of single-source SimRank search caused by two aspects. On the one hand, the iterative solution $[S^{(k)}]_{*,j}$ is used to approximate the accurate solution $[S]_{*,j}$, which leads to the iterative error. On the other hand, the dimension-reduced solution $[S_{m_1,m_2}]_{*,j}$ generated by our SimSky algorithm is used to approximate the iterative solution $[S^{(k)}]_{*,j}$, which leads in the dimension-reduced error.

Iterative Error. We analyze the iterative error. First, the same rationale as in the error analysis in Sect. 2, we can obtain that $||(\sqrt{c}A)^l|| \leq \theta\sigma^l$. Second, Lu et al. [9] proved that $\|D_k - D\| \leq c^{k+2}$. And it's obvious that $\|D\| \leq 1$. Combine the three aforementioned inequalities, we assume that $\theta = \max(1, \frac{\sigma^k}{\|(\sqrt{c}A)^k\|})$, the gap between $[S]_{*,j}$ and $[S^{(k)}]_{*,j}$ is bounded by

$$\|[S]_{*,j} - [S^{(k)}]_{*,j}\| < \theta^2 \|\sqrt{c}Ae_j\| (\frac{\sigma^{2k+1}}{1-\sigma^2} + \frac{c^{k+2}(1-\sigma^{2k+2})}{\sigma - \sigma^3}). \tag{16}$$

Example 3. Taking the column-normalised adjacency matrix A in Fig. 1 as an example, we set decay factor $c = 0.8$, scalars $\theta = 1$ and $\sigma = \rho(\sqrt{c}A) + 10^{-16}$, query node $j = 3$, number of iterations $k = 5$, the result obtained after 30 iterations is taken as the accurate solution S. By substituting these values for Eq. 16, the values on the left and right sides are 0.0996 and 1.4739. Numerical example shows that our error upper bound is feasible. □

Dimension-Reduced Error. Dimension-reduced error should be discussed separately according to the value ranges of m_1, m_2, k.

As per Lemma 3.1 in [11], for line 6 of the SimSky algorithm, we know that if $k \geq m_1+1$, only those terms $UYY_m^{i-1}e_1$ are accurate solutions to $A^i e_j$ for $1 \leq i \leq m_1$, the rest terms are approximate solutions to $A^i e_j$ for $m_1+1 \leq i \leq k$. Through the initial vector $V(:,1)$, which leads to the gap between the approximate solution and the accurate solution of the last k's terms in Eq. 11. Therefore, if there is the dimension-reduced error on the former m_1-dimensional Krylov subspace, which is transmitted to the latter m_2-dimensional Krylov subspace through the initial vector. It's difficult to give an explicit expression of the nested dimension-reduced error, so our error analysis in theory only considers $1 \leq k \leq m_1$. In the experiments, we cover all value ranges for m_1, m_2, k.

For $1 \leq k \leq m_1$ and $1 \leq k \leq m_2$, in accordance with Lemma 3.1 in [11], we know that $V(:,1)$ in line 6 and $[S_{m_1,m_2}]_{*,j}$ in line 22 are accurate solutions. Therefore, the gap between the dimension-reduced solution $[S_{m_1,m_2}]_{*,j}$ and the iterative solution $[S^{(k)}]_{*,j}$ is bounded by 0.

For $1 \leq k \leq m_1$ and $k \geq m_2+1$, according to Lemma 3.1 in [11], there is no dimension-reduced error on the former m_1-dimensional Krylov subspace, and only exists on the latter m_2-dimensional Krylov subspace. We have to establish a few auxiliary equalities to complete the analysis according to the SimSky algorithm. Due to the limited space, we ignore the specific calculation process and give a direct result. Let $k - m_2 = g$, the gap between the dimension-reduced solution $[S_{m_1,m_2}]_{*,j}$ and the iterative solution $[S^{(k)}]_{*,j}$ is bounded by

$$\|[S_{m_1,m_2}]_{*,j} - [S^{(k)}]_{*,j}\| \leq \beta h_{m_2+1,m_2}(P_1 + P_2), \tag{17}$$

where h_{m_2+1,m_2} is (m_2+1, m_2)-th entry of H, $P_1 = \sum_{i=1}^{g} \|c^i A^i\| |e_{m_2}^T H_m^{m_2+g-1-i} e_2|$, $P_2 = \sum_{i=0}^{g-1} \|c^i A^i\| \| \sum_{l=0}^{m_2+g-2-i} e_{m_2}^T H_m^{m_2+g-2-i-l} e_2 Q_l(:, 1+m_2)\|$. When $g = 0$, the equal sign "=" is established.

4 Experiments

Our experiments[2] on real datasets will evaluate the search quality of the SimSky algorithm, and verify our superiority over other competitors. We choose the optimized single-source SimRank [18] as our baseline.

4.1 Experimental Setting

Datasets. We adopt the six real datasets from the Stanford Large Network Dataset Collection[3]. They are email-Eu-core(euc), ca-GrQc(cag), Wiki-Vote(wiv), p2p-Gnutella09(p2p09), ca-AstroPh(caa) and p2p-Gnutella25(p2p25).

Metrics. To evaluate search quality, we use two metrics:

(1) *MaxError.* Given the query node j, the approximate solution $[\widetilde{S}]_{*,j}$ and the accurate solution $[S]_{*,j}$, *MaxError*$= \|[S]_{*,j} - [\widetilde{S}]_{*,j}\|_\infty = \max\{|[S]_{i,j} - [\widetilde{S}]_{i,j}|\}$ for $1 \leq i \leq n$.

(2) *Precision@k.* Given the query node j, the approximate top-k result $\widehat{V}_k = \{\widehat{v}_1, \widehat{v}_2, \cdots, \widehat{v}_k\}$, the accurate result $V_k = \{v_1, v_2, \cdots, v_k\}$, *Precision@k* $= \frac{\sum_{i=1}^{k} \delta_{\widehat{v}_i v_i}}{|\mathrm{V}_k|}$, where δ is Kronecker delta function. In our experiment, we use *Precision@500.*

Parameters. We set the decay factor $c = 0.8$. In experiments verifying Eqs. 9 and 17, we set $\theta = 1$ and $\sigma = \rho(\sqrt{c}A) + 10^{-16}$, where $\rho(\sqrt{c}A)$ represents the spectral radius of the matrix $\sqrt{c}A$.

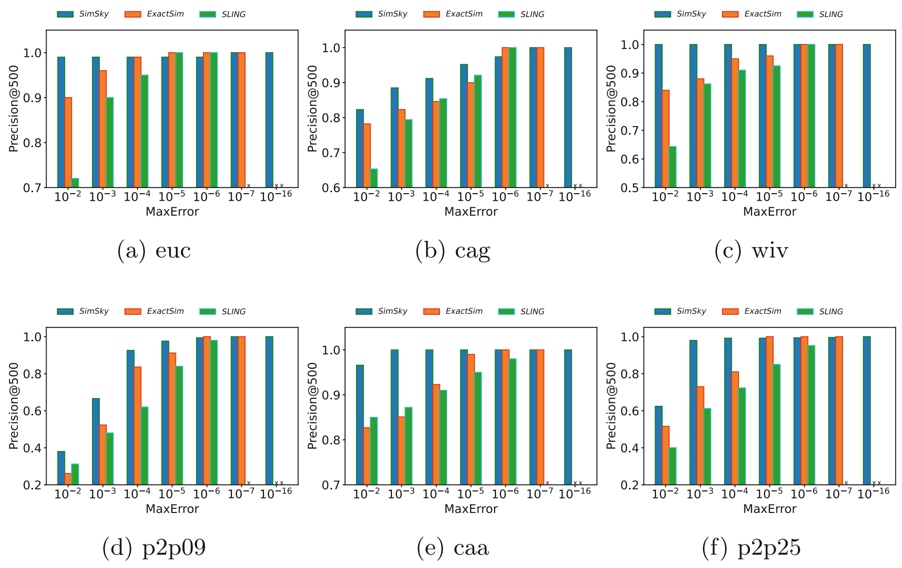
We evaluate the search quality of our SimSky algorithm and the other two competitors, including SLING [12] and ExactSim [13]. For each dataset, we generate 50 query nodes randomly and calculate their average value of *MaxError* and *Precision@500.* All experiments are run with an Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz CPU and 32 GB RAM, on windows 10.

4.2 Comparative Experiments

Our SimSky is a dimensionality reduction algorithm, SLING [12] and ExactSim [13] are random walk algorithms. To be fair, we compare their search precision and the time required under the same value of *MaxError.*

Precision. Fix $k = m_1 = m_2 = 10$, adjust the value of s, resulting in the different values of *MaxError.* We compare the precision of our SimSky with other competitors including ExactSim and SLING under the same value of *MaxError* on real datasets, as shown in Fig. 2. We notice that the ExactSim has only the ability to calculate the value of *MaxError* no less than 10^{-7} on all datasets. When the value of *MaxError* is a double-precision floating-point number, such

[2] https://github.com/AnonSimRank/SimSky.
[3] http://snap.stanford.edu/data/index.html.



(a) euc (b) cag (c) wiv

(d) p2p09 (e) caa (f) p2p25

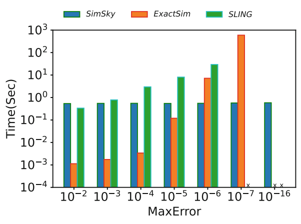
Fig. 2. Precision comparisons on all datasets

as 10^{-16}, none of our competitors are capable of doing so. However, our SimSky is able to do it within a reasonable time. Even the value of *MaxError* exceeds 10^{-6}, our SimSky attains competitive precision compared to our competitors. Especially on dataset wiv, a precision of 100% can be achieved even with the value of *MaxError* takes 10^{-2}.

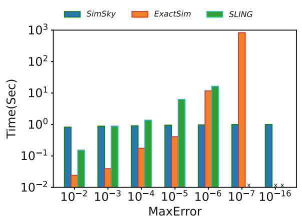
Time. Parameters are identical to the precision comparison experiments. Figure 3 depicts the cost comparisons of our SimSky with other competitors. The time required for our SimSky remains almost constant as the value of *MaxError* varies. This is consistent with our analysis, the reason lies in whatever the value of *MaxError* is, the deviation between s and n is not too big. Taking the dataset p2p09 as an example, when the values of *MaxError* are $1.0e-2$, $1.0e-3$, $1.0e-4$, $1.0e-5$, $1.0e-6$, $1.0e-7$ and $1.0e-16$, the corresponding values of s are 7600, 8000, 8085, 8094, 8101, 8102 and 8108, and $n = 8114$, so the time overhead for our SimSky doesn't vary much as the value of *MaxError* varies. For our competitors, although they require less time than our SimSky when the values of *MaxError* are $1.0e-2$ and $1.0e-3$, there is no advantage in their search precision.

4.3 Ablation Experiments

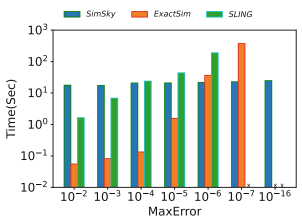
We will verify the influences of s, m_1, m_2, k on search quality and error through a series of experiments.



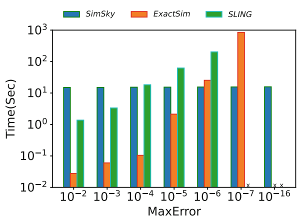
(a) euc



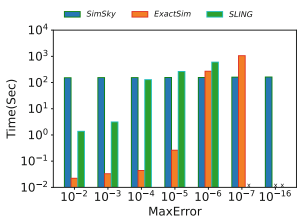
(b) cag



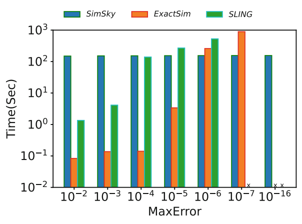
(c) wiv



(d) p2p09

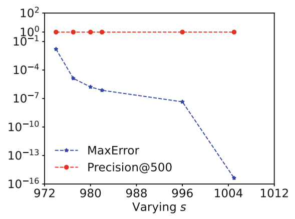


(e) caa

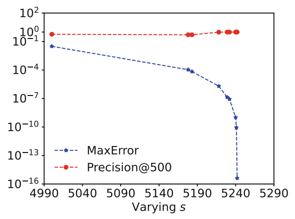


(f) p2p25

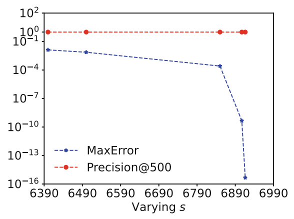
Fig. 3. Time comparisons on all datasets



(a) euc



(b) cag

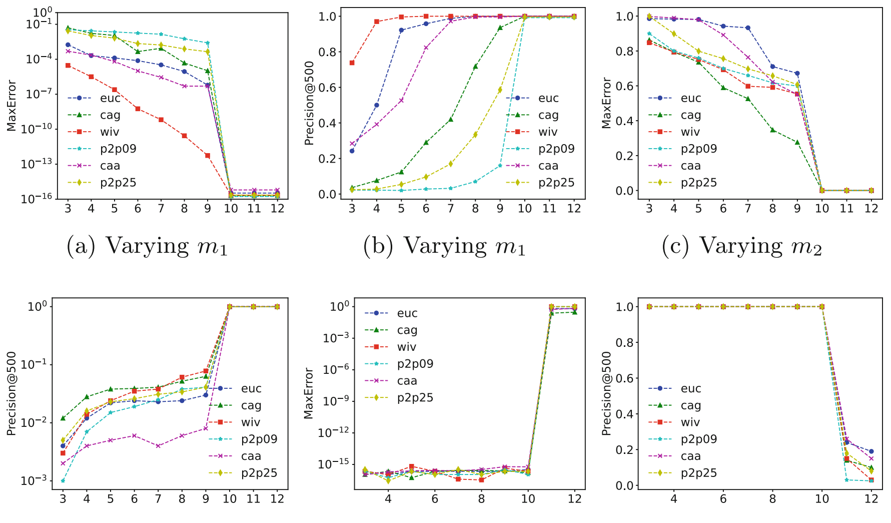


(c) wiv

Fig. 4. Effects of s on *Precision@500* and *MaxError*

Effect of s. Fix $k = m_1 = m_2 = 10$, Fig. 4 depicts the effects of s on *MaxError* and *Precision@500* on the datasets euc, cag and wiv. The gap between the dimension-reduced solution $[S_{m_1,m_2}]_{*,j}$ and the iterative solution $[S^{(k)}]_{*,j}$ narrows, as the value of s approaches the value of n. As a result, the value of error metric *MaxError* gets smaller. Instead, the value of search quality metric *Precision@500* gradually increases. And it demonstrates that our modified row orthogonal matrix W is feasible.

Effects of m_1, m_2, k. Fix $m_2 = k = 10$, $s = n$, Figs. 5a and 5b describe the influences of m_1 on *MaxError* and *Precision@500* respectively. It can be seen that the scale of the y-axis in the Fig. 5a is *logarithmic*. The value of *MaxError* shrinks as the value of m_1 approaches the value of k, which indicates that the gap between our dimension-reduced solution $[S_{m_1,m_2}]_{*,j}$ and the iterative solution $[S^{(k)}]_{*,j}$ is shrinking. It also shows that the search quality of our SimSky is



(a) Varying m_1 (b) Varying m_1 (c) Varying m_2

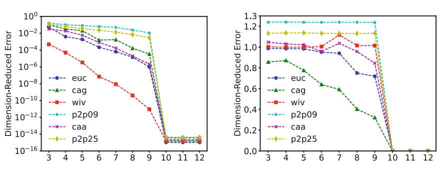
(d) Varying m_2 (e) Varying k (f) Varying k

Fig. 5. Effects of m_1, m_2, k on *Precision@500* and *MaxError*

increasing. Our dimension-reduced solution is equal to the iterative solution if the value of m_1 exceeds the value of k, so that the gap between them can be regarded as infinitesimal, and the value of precision is 1. Theoretical analysis is consistent with Fig. 5b.

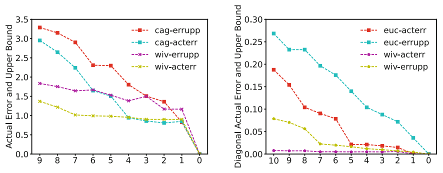
Fix $m_1 = k = 10$ and $s = n$, Figs. 5c and 5d describe the influences of m_2 on *MaxError* and *Precision@500* on all datasets respectively. It is noteworthy that the scale of the y-axis in the Fig. 5c is not *logarithmic*. Although the value of *MaxError* decreases as the value of m_2 approaches the value of k, the value of *MaxError* cannot be ignored. In other words, our SimSky is more sensitive to m_2 in comparison to m_1. This is consistent with the idea of our algorithm. Because the dimensionality of the basis matrix is n by m_1+1 on the previous m_1-dimensional Krylov subspace, but the dimensionality of the basis matrix is $(k+1)n$ by m_2+1 on the subsequent m_2-dimensional Krylov subspace. Experimental results show that the value of m_2 is not expected to be less than the value of k. When the value of m_2 exceeds the value of k, the theoretical analysis and experimental results be similar with m_1.

Fix $m_1 = m_2 = 10$ and $s = n$, Figs. 5e and 5f depict the effects of k on *MaxError* and *Precision@500* on all datasets respectively. When $k \leq 10$, SimSky returns almost the same results as the conventional iterative method. When $k > 10$, only the top-10 solutions are accurate, and the last 2 solutions are approximate in the m_1-dimensional Krylov subspace, leading to the dimension-reduced error. These results will be passed to the m_2-dimensional Krylov subspace by means of the initial vector $V(:, 1)$ in the line 6. As such, the nested dimension-reduced error cannot be ignored. This also shows that the precision of our model has been significantly affected, as shown in Fig. 5f.



(a) Varying m_1 (b) Varying m_2

Fig. 6. Effects of m_1, m_2 on error



(a) Varying $k - m_2$ (b) Varying $n - s$

Fig. 7. Actual error and upper bound

Effects of m_1, m_2 on Dimension-Reduced Error. In the experiments to verify the influences of m_1, m_2 on the dimension-reduced error, we set $s = n$ and $k = 10$, keep the remaining parameter at 10, as shown in Figs. 6a and 6b respectively. Because we use the L_2 norm of the vector to describe the dimension-reduced error, the L_∞ norm of the vector to quantify error metric *MaxError*, therefore the tendency of influence of the single variable on them should be close to consistent, as shown in Figs. 5a and 6a, 5c and 6b respectively.

Actual Error and Upper Bound. Figs. 7a and 7b depict the tendency of the actual error and its upper bound in Eqs. 17 and 9 respectively. Fix $s = n$, $m_1 = m_2 = 10$, the number of iterations k gradually decreases from 19 to 10, Fig. 7a shows that our theoretical upper bound is tight. Figure 7b depicts the tendency of actual error of the diagonal correction vector and its upper bound as the value of $n-s$ varies. When $n = s$, the values of the actual error and upper bound are 0. The theoretical analysis is consistent with the experimental result.

5 Conclusions

In this paper, we propose an accuracy-aware algorithm for efficiently computing single-source SimRank similarity. Firstly, we design an algorithm, ApproxDiag, to approximate the diagonal correction matrix with guaranteed accuracy. Secondly, we present SimSky, an algorithm that leverages two Krylov subspaces to transform high-dimensional single-source SimRank search into low-dimensional matrix-vector multiplications. To evaluate the effectiveness of SimSky, we conducted extensive experiments on various real datasets. Our results demonstrate that SimSky outperforms competing algorithms in terms of search quality.

Acknowledgments. This work has been supported by the National Natural Science Foundation of China under Grant No. 61972203.

Ethical Statement. We acknowledge the importance of ethical considerations in the design of our ApproxDiag and SimSky algorithms. All the datasets used in this paper are publicly-available online, and do not have any privacy issues. We ensure that our algorithms do not lead to any potential negative influences. We declare that we allow our algorithms to be used for the benefit of society.

References

1. Bekas, C., Kokiopoulou, E., Saad, Y.: An estimator for the diagonal of a matrix. Appl. Numer. Math. **57**(11–12), 1214–1229 (2007)
2. Boley, D.L.: Krylov space methods on state-space control models. Circuits Syst. Signal Process. **13**, 733–758 (1994)
3. Fouss, F., Pirotte, A., Renders, J.M., Saerens, M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Trans. Knowl. Data Eng. **19**(3), 355–369 (2007)
4. Fujiwara, Y., Nakatsuji, M., Shiokawa, H., Onizuka, M.: Efficient search algorithm for SimRank. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 589–600. IEEE (2013)
5. Jeh, G., Widom, J.: SimRank: a measure of structural-context similarity. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 538–543 (2002)
6. Jeh, G., Widom, J.: Scaling personalized web search. In: Proceedings of the 12th International Conference on World Wide Web, pp. 271–279 (2003)
7. Kusumoto, M., Maehara, T., Kawarabayashi, K.i.: Scalable similarity search for SimRank. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 325–336 (2014)
8. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. J. Am. Soc. Inform. Sci. Technol. **58**(7), 1019–1031 (2007)
9. Lu, J., Gong, Z., Yang, Y.: A matrix sampling approach for efficient SimRank computation. Inf. Sci. **556**, 1–26 (2021)
10. Rothe, S., Schütze, H.: CoSimRank: a flexible & efficient graph-theoretic similarity measure. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp. 1392–1402 (2014)
11. Saad, Y.: Analysis of some Krylov subspace approximations to the matrix exponential operator. SIAM J. Numer. Anal. **29**(1), 209–228 (1992)
12. Tian, B., Xiao, X.: SLING: a near-optimal index structure for SimRank. In: Proceedings of the 2016 International Conference on Management of Data, pp. 1859–1874 (2016)
13. Wang, H., Wei, Z., Yuan, Y., Du, X., Wen, J.R.: Exact single-source SimRank computation on large graphs. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 653–663 (2020)
14. Yu, W., Iranmanesh, S., Haldar, A., Zhang, M., Ferhatosmanoglu, H.: Rolesim*: scaling axiomatic role-based similarity ranking on large graphs. World Wide Web **25**(2), 785–829 (2022). https://doi.org/10.1007/s11280-021-00925-z
15. Yu, W., Lin, X., Zhang, W., Pei, J., McCann, J.A.: Simrank*: effective and scalable pairwise similarity search based on graph topology. VLDB J. **28**(3), 401–426 (2019)
16. Yu, W., McCann, J.A.: Efficient partial-pairs SimRank search on large networks. Proc. VLDB Endow. **8**(5), 569–580 (2015)
17. Yu, W., McCann, J.A., Zhang, C., Ferhatosmanoglu, H.: Scaling high-quality pairwise link-based similarity retrieval on billion-edge graphs. ACM Trans. Inf. Syst. **40**(4), 78:1–78:45 (2022). https://doi.org/10.1145/3495209
18. Yu, W., McCann, J.A.: High quality graph-based similarity search. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 83–92 (2015)

19. Yu, W., Wang, F.: Fast exact CoSimRank search on evolving and static graphs. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, 23–27 April 2018, pp. 599–608. ACM (2018). https://doi.org/10.1145/3178876.3186126
20. Yu, W., Yang, J., Zhang, M., Wu, D.: CoSimHeat: an effective heat kernel similarity measure based on billion-scale network topology. In: WWW 2022: The ACM Web Conference 2022, Virtual Event, Lyon, France, 25–29 April 2022, pp. 234–245. ACM (2022). https://doi.org/10.1145/3485447.3511952