Efficient and Accurate SimRank-based Similarity Joins: Experiments, Analysis, and Improvement [Experiment, Analysis & Benchmark]

Qian Ge[†]
Peking University
geqian@pku.edu.cn

Yu Liu[†§]
Yinghao Zhao
Yuetian Sun
Beijing Jiaotong University
{yul,yinghaozhao,yuetiansun}@bjtu.edu.cn

Lei Zou
Peking University
zoulei@pku.edu.cn

Yuxing Chen
Anqun Pan
Tencent Inc.
{axingguchen,aaronpan}@tencent.com

ABSTRACT

SimRank-based similarity joins, which mainly include threshold-based and top-k similarity joins, are important types of all-pair SimRank queries. Although a line of related algorithms have been proposed recently, they still fall short of providing approximation guarantee and suffer from scalability issues on medium and large graphs. Meanwhile, we also lack an extensive analysis of existing techniques in terms of accuracy and efficiency. Motivated by these challenges, we first conduct detailed analysis of state-of-the-art algorithms and provide additional theoretical results. Second, to address the limitations of existing techniques, we propose simple yet effective algorithm frameworks for both queries to theoretically guarantee the approximation bound, and present a more efficient all-pair algorithm inspired by randomized local push of Personalized PageRank. Next, we analyze the algorithmic complexity of threshold-based and top-k similarity joins by leveraging a reasonable assumption of SimRank distribution. Through extensive experiments, we find that our proposed methods far exceed existing ones with respect to query efficiency, approximation guarantee and practical accuracy, while our theoretical analysis nicely matches the empirical study.

PVLDB Reference Format:
Qian Ge[†], Yu Liu[†§], Yinghao Zhao, Yuetian Sun, Lei Zou, Yuxing Chen, and Anqun Pan. Efficient and Accurate SimRank-based Similarity Joins: Experiments, Analysis, and Improvement [Experiment, Analysis & Benchmark]. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code and data have been made available at https://github.com/xinghun0525/R2LP.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

†These authors contributed equally to this work.
§Corresponding author.

1 INTRODUCTION

SimRank [11] is one of the most important measures for pairwise node similarity, which finds its applications in recommender systems [18], link prediction [25], and graph embeddings [32]. The SimRank similarity of node u and v is defined by the following recursive equation, with the intuition that "two nodes in a graph are similar if their in-neighbors are similar, and a node is most similar to itself":

$$s(u,v) = \begin{cases} 1, & \text{if } u = v \\ \dfrac{c}{|I(u)||I(v)|} \displaystyle\sum_{x \in I(u)} \sum_{y \in I(v)} s(x,y), & \text{otherwise.} \end{cases} \quad (1)$$

Here, $I(u)$ denote the in-neighbors of node u, and $c \in (0,1)$ is the decay factor, which is typically set to 0.6 [31, 34, 37] or 0.8 [11]. Due to its computational complexity, efficient computation of SimRank has been extensively studied in the past two decades [7, 15, 17, 20, 22, 24, 26, 30], and remains to be a hot topic in the very recent years [21, 29, 34, 37, 45].

According to the query inputs and outputs, SimRank queries can be broadly categorized as *single-pair* queries, *single-source* queries, and *all-pair* queries (see Section 7 for more details). In particular, the all-pair query computes the estimated SimRank values for each pair of nodes (with some additive error ε). On small graphs, e.g., with tens of thousands of nodes, the *Power Method* [11] is sufficient to compute the *ground truth* SimRank values[1], which guarantee a very small absolute error ε_{min} such as 10^{-7} [34] or 10^{-11} [31]. However, this is infeasible for larger graphs because of excessive computational and memory cost. Among existing works [19, 24, 26, 27, 30, 37, 43–45] that tackle all-pair queries, state-of-the-art algorithms [37, 45] can hardly guarantee an additive error of $\varepsilon = 0.01$ on million-node graphs. Besides, a latest work [34] also indicates that it is "essentially hopeless" to compute all-pair queries on large graphs, as the output size can be as large as $O(n^2)$. Fortunately, two modified versions of all-pair queries have been proposed and studied, namely, *threshold-based* [27, 43, 44] and *top-k* [19, 30, 45] *similarity join*, of which the output sizes can be significantly smaller. Both queries

[1]We distinguish between the *exact* and the *ground truth* SimRank values, where the former is the exact solution to Equation 1 and the latter is an accurate approximation.

are more useful in practice because we are only interested in the node pairs with non-negligible similarities [45].

DEFINITION 1 (THRESHOLD-BASED SIMILARITY JOIN). *Given a graph $G = (V, E)$ and a threshold $\theta > 0$, return the set of node pairs $R(\theta)$ with $u \neq v$ and SimRank value $s(u, v) \geq \theta$.*

DEFINITION 2 (TOP-k SIMILARITY JOIN). *Given a graph $G = (V, E)$ and a parameter $k > 0$, return a k-sized set $R(k)$ containing the node pairs with the top-k largest SimRank values among all pairs $\{(u, v)|u, v \in V, u \neq v\}$.*

Following previous works [11, 34], we allow a small error parameter ε_{min} (e.g., 10^{-7}) in SimRank estimation. Considering the hardness of the problem, we additionally adopt a parameter $\rho \in (0, 1)$ named *approximation bound*.

DEFINITION 3 (APPROXIMATION BOUND). *Given an approximation parameter $\rho \in (0, 1)$, we say an algorithm $\mathcal{A}$ has approximation bound ρ for threshold-based (resp. top-k) similarity join of SimRank, if for any input graph G, the returned set of node pairs contains at least ρ fraction of the ground truth answer.*

1.1 Motivations

We note that existing algorithms for threshold-based and top-k similarity joins significantly fall short of both query efficiency and approximation guarantee. We identify the following issues.

Motivation 1. Lack of experimental analysis of efficiency and accuracy, especially on medium and large graphs. Even for state-of-the-art algorithms [37, 45] of general all-pair queries, the result accuracy is not evaluated except for small graphs due to lack of ground truth. Other representative algorithms [19, 43] even do not evaluate accuracy on small graphs.

Motivation 2. Lack of approximation guarantee for threshold-based and top-k similarity joins. Surprisingly, there does not exist an algorithm that guarantees the approximation bound for threshold-based or top-k similarity join. Algorithms with absolute error guarantee cannot be directly extended to achieve this goal.

Motivation 3. Lack of understanding of the computational complexity. To the best of our knowledge, it is unclear how the input parameters and graph properties impact the algorithmic complexity of both queries. The complexity analysis for general all-pair algorithms does not reveal the intrinsic hardness of threshold-based and top-k similarity joins.

1.2 Contributions

For both types of all-pair SimRank queries, we analyze state-of-the-art algorithms in detail, provide improved algorithm with approximation guarantee and lower theoretical complexity, and discuss the problem complexity w.r.t. the input parameters and graph properties. Our main contributions are summarized as follows.

Detailed analysis of state-of-the-art algorithms. We choose four representative algorithms, *UISim* [45], *FLP & Opt-LP* [37], *H-go SRJ* [43], and *KSimJoin* [19] for comparison. *UISim* and *FLP & Opt-LP* are state-of-the-art algorithms for general all-pair queries, which are directly extended to answer SimRank-based similarity joins [45]. *H-go SRJ* and *KSimJoin* are the state-of-the-art algorithms for threshold-based and top-k similarity joins, respectively. For each algorithm, we discuss its basic idea, theoretical guarantee, complexity analysis and empirical study. Specifically, we propose additional theoretical results (w.r.t. complexity or error guarantee) of these algorithms that are incorrect or missed in the original papers, and discuss their limitations in answering similarity join queries.

Improved algorithms for SimRank-based similarity joins. We first propose two simple yet effective algorithm frameworks for both queries respectively with approximation bound. The basic idea is to invoke an efficient algorithm for all-pair queries that estimates SimRank values within ε error, and to gradually shrink ε until the approximation bound is theoretically guaranteed. Next, we utilize the equivalence of all-pair SimRank on the input graph G and single-target Personalized PageRank (PPR) on the *SimRank graph* G^s [11, 37], and devise Randomized Reverse Local Push (*R^2LP*) inspired by randomized backward push for PPR [33]. It improves the all-pair query complexities from $O(\frac{\sum_{u,v \in V} d_{in}(u) d_{in}(v) s(u,v)}{\varepsilon})$ (the best known algorithm [37]) to $\tilde{O}(\frac{\sum_{u,v \in V} \sqrt{d_{in}(u) d_{in}(v)} s(u,v)}{\varepsilon})$. We further propose a pruning strategy to dramatically enhance practical efficiency while retaining error guarantee. Our empirical study shows that *R^2LP* significantly improves query efficiency for general all-pair queries and similarity joins compared to existing methods.

Complexity analysis by utilizing SimRank distribution. We find that it is almost impossible to give a non-trivial complexity bound for both similarity join queries without any assumption of SimRank distribution. On the other hand, our empirical analysis validates that for most real-world graphs, the ground truth SimRank values follow some *generalized* version of power-law distribution [2, 5], which is consistent with previous studies [21, 34]. We leverage this property to propose verifiable assumptions of SimRank distribution and present complexity analysis for our algorithms, which serves as the upper bound of the algorithmic complexity of SimRank-based similarity joins. Our theoretical results nicely match the empirical study in Section 6.

Extensive experiments on medium and large graphs. We conduct extensive experiments on small, medium, and large sized graphs and compare our algorithms with state-of-the-art methods. As far as we know, we first employ medium and large datasets to systematically evaluate threshold-based and top-k similarity joins in terms of running time, absolute error and query accuracy, and demonstrate the necessities of holding approximation bound as well as devising more efficient algorithms. Experimental study demonstrates that our algorithms outperform existing ones in most cases, achieving better accuracy while being faster by up to an order of magnitude. More importantly, our empirical findings validate that it is the skewness of SimRank distribution that mainly determines the problem hardness. This results verifies the effectiveness of our theoretical analysis.

2 PRELIMINARIES

The definition in Equation 1 can be interpreted via the *node-pairs graph* G^2 [11]. Each node in G^2 represents a pair of nodes (u, v) in G. If there exist edges (u, v) and (x, y) in G, edge $((u, x), (v, y))$ is included in G^2. Specifically, every node (v, v) is called the *singleton node*. To this end, SimRank can be thought of as propagating the similarity among nodes in G^2, starting from all the singleton nodes.

Matrix formation of SimRank. The recursive definition can be represented in matrix formation [42], i.e., $S = cP^\intercal SP \vee I$, where S is an $n \times n$-dimensional similarity matrix with $S[u, v] = s(u, v)$,

Table 1: Table of notations.

Notation	Description
$G = (V, E)$	A graph with node set V and edge set E, where $n = \|V\|$ and $m = \|E\|$
G^2, G^s	The node-pairs graph [11] and the SimRank graph [37]
u, v	Nodes in G
$I(u), O(u)$	The incoming and outgoing neighbors of u, and we have $d_{in}(u) = \|I(u)\|, d_{out}(u) = \|O(u)\|$
$s(u, v)$	The SimRank value, i.e., the solution of Equation 1
$s^*(u, v)$	The ground truth SimRank value, with $\|s^*(u, v) - s(u, v)\| \leq \varepsilon_{min}$ (with high probability)
$\hat{s}(u, v)$	The estimated SimRank value
ε, δ	The additive error and the failure probability
θ	The input parameter of threshold-based similarity join
k	The input parameter of top-k similarity join
ρ	The approximation bound

P is the column-normalized adjacency matrix of G, I is the identity matrix, and operator $\vee$ denotes element-wise maximum. The all-pair ground truth of SimRank can be computed by the *Power Method* [11], which is essentially a fixed-point iteration process on G^2. Recent literature [17, 24, 37] further derives various close-form solutions based on this formation.

The random walk interpretation. Jeh and Widom explain the SimRank definition from random walk perspective in their original paper [11]. We adopt $\sqrt{c}$-walk, its revised version following [12, 20, 31]. Let $W_{\sqrt{c}}(u)$ denote a random walk starting from node u and following incoming edges at each step with walking probability $\sqrt{c}$. More precisely, at each step, with probability $\sqrt{c}$, it randomly chooses an in-neighbor and continues the walk, otherwise the walk terminates at the current node. [31] proves that the SimRank of u and v equals the probability that two $\sqrt{c}$-walks $W_{\sqrt{c}}(u)$ and $W_{\sqrt{c}}(v)$ meet.

Table 1 lists the frequently used notations throughout the paper.

3 ANALYSIS OF STATE OF THE ART

We present detailed comparison and analysis of state-of-the-art algorithms for threshold-based and top-k similarity joins. Specifically, we choose *UISim* [45], *FLP & Opt-LP* [37], *H-go SRJ* [43] and *KSimJoin* [19] because they have not been outperformed or evaluated by previous empirical analysis. We illustrate their basic ideas in Figure 1, and summarize the theoretical guarantee, time complexity, and previous empirical study in Table 2.

3.1 *UISim*

The algorithm depends on the random walk interpretation of SimRank and adopts a *hub-based strategy* to prioritize all possible random walk pairs. First, the hub set H is constructed from the top-$|H|$ nodes with the largest *in-degrees*. We say an l-length random walk $W(u) = (p_1 = u, p_2, \ldots, p_l)$ from u has *hub length* k if $|H \cap \{p_2, \ldots, p_{l-1}\}| = k$. Intuitively, large hub length implies small walking probability. Second, the SimRank value of two nodes u and v is decomposed according to the hub length:

$$\hat{s}^{(\eta)}(u, v) = \sum_{k=0}^{\eta} \sum_{\max(i,j)\leq k} R(P_u^i \bowtie P_v^i), \tag{2}$$

where η denotes the maximum hub length, P_u^i denotes the set of random walks that start from u and are of hub length i, and the join operator assembles pairs of random walks from P_u^i and P_v^i that have the same walk length and meet at the same node. Given the set of assembled walk pairs, the $R(\cdot)$ operator returns the sum of their meeting probabilities. As demonstrated in Equation 2, *UISim* assembles random walks in ascending order of hub length to prioritize them roughly by the meeting probabilities.

In practice, *UISim* introduces the notion of *prime subgraphs* and leverages graph expansion to estimate SimRank values incrementally (Figure 1(a)). The prime in-subgraph (resp. out-subgraph) of node v contains all reachable nodes starting from v and following in-edges (resp. out-edges), until a hub node is encountered. For all-pair SimRank query, *UISim* first considers the prime out-subgraphs of all nodes with at least two out-neighbors (denoted as $V_{\geq 2}^o$), and assembles all possible pairs of random walks following Equation 2 (with $k = 0$). Then, each subgraph is expanded with the prime out-subgraphs of the encountered hub nodes, which corresponds to increasing k by one. The algorithm terminates when k reaches η.

Theoretical Guarantee. *UISim* relaxes the first-meeting constraints of SimRank definition[2], and conducts analysis of the expected error bound based on this assumption. Generally, it does not guarantee an absolute error of ε for SimRank estimation $\hat{s}(u, v)$, which says $|\hat{s}(u, v) - s(u, v)| \leq \varepsilon$.

Complexity Analysis. According to [45], the time complexity of *UISim* for all-pair queries is bounded by $O(|V_{\geq 2}^o||H|^\eta T)$ in expectation, where $T = O\left(d^L\left(1 - \sum_{v\in H} d_v/2m\right)^L\right)$ stands for the expected number of random walks up to length L. It is assumed that every prime subgraph has $O(|H|)$ hubs.

Empirical Analysis. *UISim* is evaluated for all-pair queries (in terms of absolute error) and for top-k similarity joins (in terms of Precision@k). However, the evaluation is only conducted on small graphs with tens of thousands nodes.

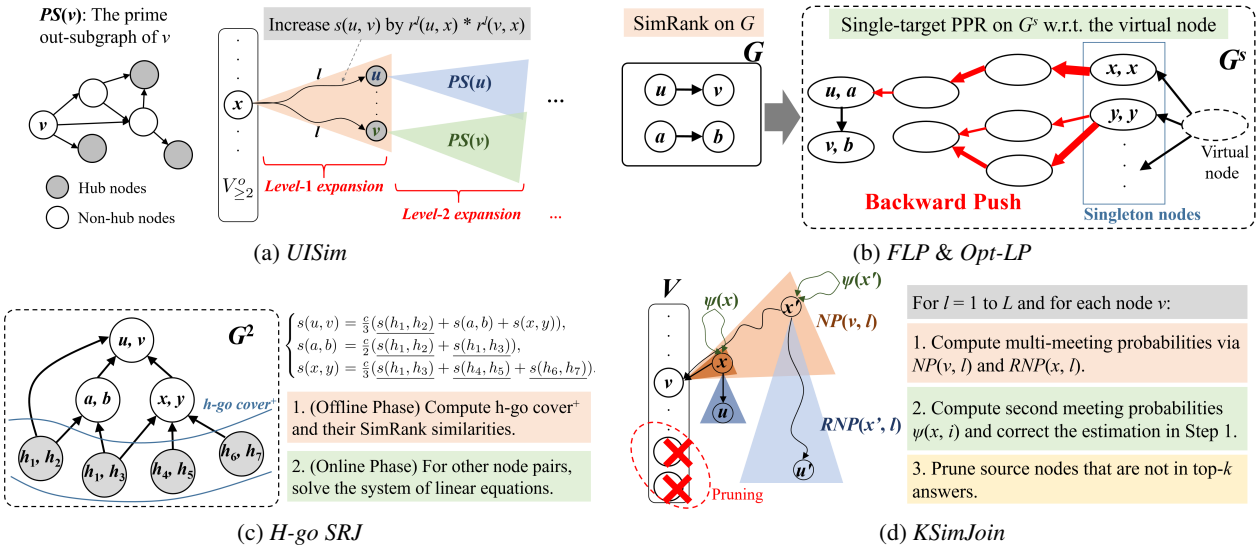
3.2 *FLP & Opt-LP*

Motivated by a modified version of the node-pairs graph, namely, the *SimRank graph*, Wang et al. [37] propose a new linear system of SimRank with the Kronecker product and vector operators:

$$[I - c(I - E)(P^\intercal \otimes P^\intercal)]\overrightarrow{vec}(S) = \overrightarrow{vec}(I), \tag{3}$$

where $E = Diag(\overrightarrow{vec}(I))$, $\otimes$ denotes the Kronecker product, S is the SimRank matrix with $S[u, v] = s(u, v)$, $\overrightarrow{vec}(\cdot)$ transforms an $N \times N$-dimensional matrix to an N^2-dimensional vector by concatenating its columns, and $Diag(\cdot)$ transforms an N'-dimensional vector to an $N' \times N'$-dimensional diagonal matrix. Based on that, a local push algorithm *ForwardLocalPush (FLP)* is proposed. The authors further devise a reduced linear system to avoid the redundant computation of both $s(u, v)$ and $s(v, u)$, and to effectively speed up the computation involving self-loops. The optimized algorithm is referred to as *OptimizeLocalPush (Opt-LP)*. We propose a slightly modified version of the SimRank graph [37].

[2]Note that [45] provides a correction method to exclude *multi-hop double-meeting probabilities*, i.e., the probability of two random walks meeting exactly twice. It is non-trivial to extend the method to *multi-meeting probabilities*.



(a) *UISim* (b) *FLP & Opt-LP*

(c) *H-go SRJ* (d) *KSimJoin*

Figure 1: Overview of state-of-the-art algorithms for SimRank-based similarity joins.

Table 2: Comparison of state of the art. N/A means that the specific query is not studied in the original paper.

Algorithm	Theoretical Guarantee			Time Complexity	Empirical Study		
	Absolute	Threshold	Top-k		Absolute	Threshold	Top-k
UISim [45]	✗	N/A	✗	$O(\|V^o_{\geq 2}\|\|H\|^\eta T)$	✓	N/A	✓
FLP & Opt-LP [37]	✓	N/A	N/A	$O(\frac{\sum_{u,v\in V} d_{in}(u)d_{in}(v)s(u,v)}{\varepsilon})$	✓	N/A	N/A
H-go SRJ [43]	✓	✗	N/A	$\tilde{O}(n\bar{d}^h + \frac{n\|H(G)\|}{\varepsilon^2})$ (offline), $O(n\bar{d}^{h+3})$ (online)	N/A	✓(only query time)	✓(only query time)
KSimJoin [19]	✓	N/A	✗	$O(n\bar{d}^{\log\frac{1}{\varepsilon}})$	N/A	N/A	✓(only query time)

DEFINITION 4 (SIMRANK GRAPH). *Given a graph G, its SimRank graph G^s is obtained from the node-pairs graph G^2 by removing all the in-edges of the singleton nodes. Besides, we modify G^s by adding a virtual node (v_r, v_r), and for each singleton node (v, v), we add one in-edge from it to the virtual node.*

Noticing the close relation of SimRank to PPR, [37] claims that the algorithm is essentially the *Backward Push* [23] for single-target PPR queries, except that the PPR is defined following in-edges rather than out-edges, which is referred to as the *reverse* PPR (Figure1(b)). In practice, the algorithm does not have to materialize G^s, and we demonstrate it in Figure 1(b) as the dashed rectangle. We reformalize the claim in [37] as follows. For space constraint, we refer all proofs to the full version of the paper [1].

LEMMA 1. *The SimRank similarity of u and v is exactly $\frac{1}{c(1-c)}$ times the reverse PPR value of node (u, v) w.r.t. the target node (v_r, v_r) on SimRank graph G^s and with stopping probability $1-c$:*

$$s(u,v) = \frac{1}{c(1-c)}\pi_{G^s}((u,v),(v_r,v_r)). \quad (4)$$

Theoretical Guarantee. *FLP & Opt-LP* achieves absolute error guarantee following *Backward Push*. The algorithm takes an error parameter ε as input, and guarantees $|\hat{s}(u,v) - s(u,v)| \leq \varepsilon$ for each node pair (u, v).

Complexity Analysis. We observe that the proposed *average* time complexity $O(\frac{\bar{d}^2}{\varepsilon})$ (Proposition 1 of [37]) is *incorrect* where $\bar{d}$ denotes the average degree of G. As with *Backward Push* [23], the average complexity is achieved by considering all possible target nodes in the graph. However, as Lemma 1 shows, the target node is *fixed* as the virtual node. We fix the issue using the following lemma.

LEMMA 2. *To achieve an absolute error of ε for all-pair queries, following [23, 33], the time complexity of FLP and Opt-LP are bounded by $O(\frac{\sum_{u,v\in V} d_{in}(u)d_{in}(v)s(u,v)}{\varepsilon})$.*

Empirical Analysis. [37] conducts experiments for all-pair queries on both small and large graphs with respect to query time, but only evaluates estimation error on small graphs with thousands of nodes. Due to the deficiency of *Backward Push* and as our experiment shows, the algorithms have scalability issues on million-node graphs.

3.3 *H-go SRJ*

H-go SRJ [43] is an index-based algorithm for threshold-based similarity joins. Inspired by the distance-based upper bound of SimRank, it presents an index called *h-go cover*$^+$ for the node-pairs graph G^2, so that for any node pair, the SimRank value is either recorded by the index or can be recovered from the indexed neighbors with at most h-hops. As shown in Figure 1(c), in the offline phase, the

algorithm first computes an h-go cover$^+$ without materializing the node-pairs graph. For each indexed node pair, its SimRank value can be estimated by a single-pair query. In the online phase, for every non-indexed node pair, *H-go SRJ* first generates an extending tree rooted at this node pair, then builds a system of linear equations following the recursive definition of SimRank and with the help of the index. The equations can be solved using Gaussian elimination.
Theoretical Guarantee. [43] does not discuss the error guarantee of *H-go SRJ*. We remedy this issue with the following analysis. Given an error parameter ε, to estimate each indexed node pair with absolute error guarantee, we invoke the Monte Carlo algorithm to sample $\tilde{O}(\frac{1}{\varepsilon^2})$ pairs of random walks (the $\tilde{O}$ notation ignores log factors). The following lemma bounds the estimation error of the online phase.

LEMMA 3. *If all indexed SimRank similarities have at most ε additive error, then this ε error guarantee also holds for the similarities of any node pair computed in the online phase.*

Complexity Analysis. According to [43], the h-go cover$^+$ selection takes $O(n\bar{d}^h)$ time where $\bar{d}$ is the average degree of G. We notice that computing SimRank estimation for the indexed node pairs takes an extra $\tilde{O}(\frac{n|H(G)|}{\varepsilon^2})$ time, where $H(G)$ denotes the h-go cover$^+$ of G. Therefore, the offline phase needs $\tilde{O}(n\bar{d}^h + n\frac{|H(G)|}{\varepsilon^2})$ time. For the online phase, given a node pair, generating the extending tree needs $O(\bar{d}^{2h})$ time, while solving the linear equations incurs $O(\bar{d}^{2h+1})$ time with all optimizations [43]. Since the linear system contains $O(\bar{d}^{2(h-1)})$ unknown values on average, the amortized cost for computing each non-indexed node pair is $O(\bar{d}^3)$. Assume there exist $O(n\bar{d}^h)$ non-indexed node pairs after distance-based pruning, the online phase needs $O(n\bar{d}^h \cdot \bar{d}^3) = O(n\bar{d}^{h+3})$ time.
Empirical Analysis. The authors of *H-go SRJ* conduct experiments for query efficiency on graphs with up to millions of nodes. However, they do not evaluate the accuracy of the proposed method.

3.4 *KSimJoin*

Li et al. [19] propose an incremental framework for top-k similarity joins based on the random walk interpretation (Figure 1(d)). The key idea is to first compute all meeting probabilities and then exclude the multi-meeting cases. In particular, it decomposes the SimRank value by meeting steps: $s(u,v) = \sum_{l=0}^{\infty} s^{(l)}(u,v)$. Here, $s^{(l)}(u,v)$ is the first meeting probability of two $\sqrt{c}$-walks from u and v at exact l-th step. To estimate $s^{(l)}(u,v)$, the following equation is adopted:

$$s^{(l)}(u,v) = \sum_{x \in V} \Pr((u,v) \xrightarrow{l} (x,x)) - \sum_{i \in [1,l-1], y \in V} \Pr((u,v) \xrightarrow{i} (y,y)) \psi(y, l-i), \quad (5)$$

where $\psi(y, l-i)$ is the probability of two $\sqrt{c}$-walks starting from y first meeting at exact $(l-i)$-th step, which is referred to as the *second meeting probability*.

To implement the above idea, the *KSimJoin* algorithm estimates $s^{(l)}(u,v)$ in ascending order of l for $l \in [1, L]$. In l-th iteration, for every node v in the remaining node set R (where R is initialized as V), it first performs a breadth-first search (BFS) from v following in-edges. All visited node at step l is referred to as $NP(v,l)$. Then, for each node $x \in NP(v,l)$, a BFS following out-edges is conducted to find all node u that is l-step from u, denoted as $RNP(x,l)$. The multi-meeting probabilities are computed from $NP(v,l)$ and $RNP(x,l)$ for all $x \in NP(v,l)$. By leveraging Equation 5, it excludes all multi-meeting cases, where the second meeting probabilities $\psi(x,i)$ for each x and $i \in [1, l-1]$ can be computed following a similar BFS-based and layer-by-layer iteration approach. For top-k queries, after each iteration of l, *KSimJoin* uses a carefully designed upper bound to prune nodes in R that cannot be the top-k answers. The upper bound is computed by relaxing Equation 5.
Theoretical Guarantee. As [19] does not analyze the error guarantee of *KSimJoin*, we formally present the following lemma, which states that *KSimJoin* is essentially equivalent to the *Power Method*.

LEMMA 4. *By setting $L = O(\log \frac{1}{\varepsilon})$, KSimJoin achieves $O(\varepsilon)$ absolute error guarantee.*

Complexity Analysis. Computing the multi-meeting probabilities and the upper bound incurs $O(|R|\bar{d}^{2L})$ time [19]. The cost of estimating the second meeting probabilities is asymptotically identical. Since $L = O(\log \frac{1}{\varepsilon})$ and $|R| = O(n)$, the overall time complexity is $O(n\bar{d}^{\log \frac{1}{\varepsilon}})$.
Empirical Analysis. Empirical study of [19] only focuses on the efficiency of top-k queries on small and medium graphs.

3.5 Comparison of State-of-the-art Algorithms

H-go SRJ and *FLP & Opt-LP* rely on the recursive SimRank definition while *UISim* and *KSimJoin* adopt the random walk interpretation. To compute the meeting probabilities, *KSimJoin* uses breadth-first search, *FLP & Opt-LP* depend on local push algorithms, while *UISim* is considered a tradeoff between them. Moreover, we derive a few conclusions. First, their theoretical complexities are not comparable. Particularly, [43, 45] introduce parameters defined by the algorithms. Second, existing experimental study is quite limited, including lack of evaluation on large graphs [19, 37, 45] and for estimation error [19, 43]. Third, although some algorithms achieve absolute error guarantee or effectively prune a fraction of unpromising nodes, none of them provide approximation guarantees for both types of similarity joins. For example, as shown by our experiments, threshold-based similarity joins with parameter θ cannot be directly answered by estimating node pairs with $O(\theta)$ error.

4 OUR IMPROVED ALGORITHMS

4.1 Algorithm Framework

We adopt a simple yet effective idea to answer threshold-based and top-k queries with approximation bound. It takes an all-pair algorithm $\mathcal{AP}$ as subprocedure, and progressively decreases the input error parameter of $\mathcal{AP}$ until the bound is satisfied. Though similar idea has been used to answer single-source top-k queries for PPR [35] and SimRank [21], we present specific stopping conditions for our problems.

4.1.1 APThres for Threshold Queries.

Recall that threshold-based similarity join finds (at least ρ fraction of) all node pairs with SimRank similarity above θ. Given an all-pair algorithm that guarantees an additive error of ε (with high probability), i.e., $|\hat{s}(u,v) -$

$s(u,v)| \leq \varepsilon$ for each node pair (u,v), if the estimated SimRank value $\hat{s}(u,v)$ satisfies that $\hat{s}(u,v) \geq \theta + \varepsilon$, then we have $s(u,v) \geq \theta$, that is, (u,v) is included in the result set. Similarly, for (u',v') with $\hat{s}(u',v') < \theta - \varepsilon$, we should exclude it from the result set. Intuitively, ε should be small enough to meet the approximation bound ρ. We iteratively decrease ε by half until the condition is satisfied.

Algorithm Description. The framework for threshold-based queries, denoted as *APThres*, is illustrated in Algorithm 1. It takes graph G, threshold θ, approximation bound ρ, and an all-pair algorithm $\mathcal{AP}$ with absolute error guarantee as input. The algorithm initializes the result set $\mathcal{R}(\theta)$ and the candidate set $\mathcal{C}$ as empty set, and set $\varepsilon_1 = O(\theta)$ (e.g. $\frac{\theta}{2}$) for the first iteration (Line 1). Then, at i-th iteration, *APThres* iteratively invokes $\mathcal{AP}$ with error parameter ε_i (Lines 2-3). Particularly, we use $\hat{S}_i = \{\hat{s}_i(u,v) | u,v \in V, u \neq v\}$ to denote the all-pair SimRank estimations, where the subscript i specifies the iteration number. In practice, we do not store node pairs with zero estimated values. We use $\hat{S}_i$ to update $\mathcal{R}(\theta)$ (Line 4) and $\mathcal{C}$ (Line 5). The algorithm terminates if $\frac{|\mathcal{R}(\theta)|}{|\mathcal{R}(\theta)|+|\mathcal{C}|} \geq \rho$ holds, which implies that $\mathcal{R}(\theta)$ contains at least ρ fraction of the result (Lines 6-7). Otherwise, we set the error parameter of the next iteration as the half of its current value (Line 8). Note that if our stopping condition is never satisfied, the algorithm will stop when the SimRank estimation of every node pair has no more than ε_{min} error. In both cases, $\mathcal{R}(\theta)$ is returned (Line 9).

Algorithm 1: The *APThres* framework

Input: Directed graph $G = (V, E)$, threshold θ, approximation bound ρ, all-pair algorithm $\mathcal{AP}$
Output: $\mathcal{R}(\theta)$, which contains at least ρ fraction of all node pairs with similarities above θ

1 $\mathcal{R}(\theta) \leftarrow \emptyset, \mathcal{C} \leftarrow \emptyset, \varepsilon_1 \leftarrow O(\theta)$;
2 **for** $i = 1$ *to* $\log_2 \frac{1}{\varepsilon_{min}}$ **do**
3 $\quad \hat{S}_i \leftarrow \mathcal{AP}(G, \varepsilon_i)$;
4 $\quad \mathcal{R}(\theta) \leftarrow \mathcal{R}(\theta) \cup \{((u,v), \hat{s}_i(u,v)) \in \hat{S}_i | \hat{s}_i(u,v) \geq \theta + \varepsilon\}$;
5 $\quad \mathcal{C} \leftarrow (\mathcal{C} \cup \{((u,v), \hat{s}_i(u,v)) \in \hat{S}_i | \hat{s}_i(u,v) \in [\theta - \varepsilon, \theta + \varepsilon)\}) \backslash \mathcal{R}(\theta)$;
6 $\quad$ **if** $\frac{|\mathcal{R}(\theta)|}{|\mathcal{R}(\theta)|+|\mathcal{C}|} \geq \rho$ **then**
7 $\quad\quad$ **break**;
8 $\quad \varepsilon_{i+1} \leftarrow \varepsilon_i / 2$;
9 **return** $\mathcal{R}(\theta)$;

Correctness. The following lemma proves that *APThres* has approximation bound for threshold-based queries.

LEMMA 5. *APThres with an all-pair algorithm $\mathcal{AP}$ of absolute error guarantee holds approximation bound ρ for threshold-based similarity joins.*

4.1.2 APTop-k for Top-k Queries. We provide a framework *APTop-k* for top-k similarity joins with approximation bound ρ, which is analogous to *APThres* but with different stopping condition.

Algorithm Description. We demonstrate the pseudocode in Algorithm 2. The algorithm set the initial value of the error parameter as $O(c)$ (Line 1), and iteratively invokes the all-pair algorithm (Lines 2-3). At i-th iteration, the estimation values $\hat{S}_i$ are sorted in descending order, denoted as $\{\hat{s}_{i,1}, \hat{s}_{i,2}, \ldots, \hat{s}_{i,k}, \ldots, \hat{s}_{i,n(n-1)}\}$ (Line 4). We set the stopping conditions as $\hat{s}_{i,\lceil \rho k \rceil} - \varepsilon_i \geq \hat{s}_{i,k+1} + \varepsilon_i$ (Lines 5-6), which says that the lower bound of the $\lceil \rho k \rceil$-th largest estimation is no smaller than the upper bound of the $(k+1)$-th one. The algorithm decreases the error parameter by half if it does not terminate this round (Line 7). Finally, *APTop-k* returns the set of node pairs with the top-k largest estimations (Lines 8-9).

Algorithm 2: The *APTop-k* framework

Input: Directed graph $G = (V, E)$, parameter k, approximation bound ρ, all-pair algorithm $\mathcal{AP}$
Output: $\mathcal{R}(k)$, which contains at least ρ fraction of node pairs with top-k largest SimRank values

1 $\varepsilon_1 \leftarrow O(c)$;
2 **for** $i = 1$ *to* $\log_2 \frac{1}{\varepsilon_{min}}$ **do**
3 $\quad \hat{S}_i \leftarrow \mathcal{AP}(G, \varepsilon_i)$;
4 $\quad \{\hat{s}_{i,1}, \hat{s}_{i,2}, \ldots, \hat{s}_{i,k}, \ldots, \hat{s}_{i,n(n-1)}\} \leftarrow \text{sort}(\hat{S}_i)$ (in descending order);
5 $\quad$ **if** $\hat{s}_{i,\lceil \rho k \rceil} - \varepsilon_i \geq \hat{s}_{i,k+1} + \varepsilon_i$ **then**
6 $\quad\quad$ **break**;
7 $\quad \varepsilon_{i+1} \leftarrow \varepsilon_i / 2$;
8 Let $\mathcal{R}(k)$ be the node pairs with top-k largest estimations;
9 **return** $\mathcal{R}(k)$;

Correctness. We have the following lemma for *APTop-k*.

LEMMA 6. *APTop-k with an all-pair algorithm $\mathcal{AP}$ of absolute error guarantee holds approximation bound ρ for top-k queries.*

Remark. One might speculate that iteratively invocation of the all-pair algorithm $\mathcal{AP}$ in *APThres* and *APTop-k* is costly and can be improved by the following strategy. When the size of the candidate set $\mathcal{C}$ is below some threshold (note that we can retain $\mathcal{C}$ for *APTop-k* similar to *APThres*), or the estimated cost of handling $\mathcal{C}$ is less than invoking $\mathcal{AP}$, we switch to estimate the SimRank values of node pairs in $\mathcal{C}$ independently. Actually, we have tried this strategy but always find it having inferior query efficiency. We anticipate that it may be useful for partial-pair queries (especially when the query node set is small), however, a fully investigation of this setting is beyond the scope of this paper.

4.2 Randomized Reverse Local Push (R^2LP)

We present an all-pair algorithm with better theoretical complexity, which is inspired by the advantage of *Randomized Backward Search (RBS)* [33] over *Backward Push* for PPR. To guarantee the practical efficiency, we devise a pruning strategy with absolute error guarantee, which dramatically enhances query time.

Algorithm Description. We refer to the algorithm as *Randomized Reverse Local Push* (R^2LP) (see Algorithm 3), which naturally extends the *RBS* algorithm [33] for PPR to all-pair SimRank queries. Specifically, the SimRank value $s(u,v)$ is split into $s^{(l)}(u,v)$ for $l \in [0, \infty)$, where $s^{(l)}(u,v)$ is the probability of two $\sqrt{c}$-walks $W_{\sqrt{c}}(u)$ and $W_{\sqrt{c}}(v)$ first meet after exact l steps. It is obvious that $s(u,v) = \sum_{l=0}^{\infty} s^{(l)}(u,v)$. R^2LP estimates $s^{(l)}(u,v)$ for all node pairs in ascending order of l. To guarantee ε additive error, it is sufficient to consider $l \leq L = O(\log \frac{1}{\varepsilon})$ (Line 1). We initialize

the algorithm in Lines 2-3, where $s^{(l)}(u,v)$ corresponds to the reserves of PPR [23, 33], and we omit the residues because it holds that $r^{(l)}(u,v) = s^{(l)}(u,v)/(1-c)$. Then, for $l = 0$ to $L-1$, we compute the estimates $s^{(l+1)}(u,v)$ based on $s^{(l)}(u,v)$ (Lines 4-10). For each $s^{(l)}(u,v)$ at level l, we check if its value is large enough (Line 5). For now, we set $f(c,\varepsilon) = 0$ to be the same as *RBS*. Similarly, our algorithm checks each out-neighbor (u',v') of (u,v) in the SimRank graph G^s (Lines 6&9). According to the values of $s^{(l)}(u,v), d_{in}(u'), d_{in}(v')$, and ε, the algorithm either pushes deterministically (Line 7) or randomly (Lines 8&10). The returned value $\hat{s}(u,v)$ summarizes the estimations of every step l (Lines 11-12).

Algorithm 3: Randomized Reverse Local Push (R^2LP)

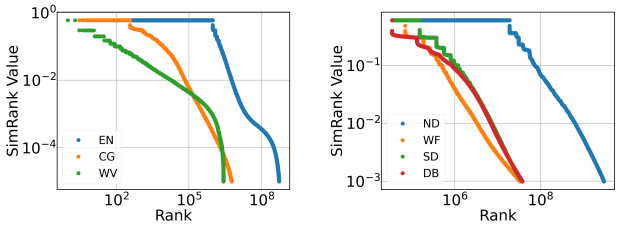
Input: Directed graph $G = (V,E)$ with sorted adjacency lists, additive error parameter ε
Output: $\hat{s}(u,v)$ for all $u,v \in V$

1 $L \leftarrow \log_c(1-c)\varepsilon$;
2 $\hat{s}^{(l)}(u,v) \leftarrow 0$ for $l \in [0,L]$ and $u,v \in V$;
3 $\hat{s}^{(0)}(v,v) \leftarrow 1$ for all $v \in V$;
4 **for** $l = 0$ *to* $L-1$ **do**
5 **for** *each* (u,v) *with* $\hat{s}^{(l)}(u,v) > f(c,\varepsilon)$ **do**
6 **for** *each* $u' \in O(u), v' \in O(v), u' \neq v'$ *and* $\sqrt{d_{in}(u')d_{in}(v')} \leq \frac{c\hat{s}^{(l)}(u,v)}{(1-c)\varepsilon}$ **do**
7 $\hat{s}^{(l+1)}(u',v') \leftarrow \hat{s}^{(l+1)}(u',v') + \frac{c\hat{s}^{(l)}(u,v)}{d_{in}(u')d_{in}(v')}$;
8 $r \leftarrow rand(0,1)$;
9 **for** *each* $u' \in O(u), v' \in O(v), u' \neq v'$ *and* $\frac{c\hat{s}^{(l)}(u,v)}{(1-c)\varepsilon} < \sqrt{d_{in}(u')d_{in}(v')} \leq \frac{c\hat{s}^{(l)}(u,v)}{(1-c)\varepsilon r}$ **do**
10 $\hat{s}^{(l+1)}(u',v') \leftarrow \hat{s}^{(l+1)}(u',v') + (1-c)\varepsilon$;
11 $\hat{s}(u,v) \leftarrow \sum_{l=0}^{L} \hat{s}^{(l)}(u,v)$;
12 **return** non-zero $\hat{s}(u,v)$s for $u,v \in V, u \neq v$;

We discuss two implementation details. First, we do not need to materialize G^s. To get all out-neighbors of (u,v) in G^s, it is sufficient to traverse the out-neighbors of u and v in G simultaneously. Second, to guarantee the efficiency, the randomized push strategy [33] needs the out-neighbors of (u,v) to be sorted in *ascending* order of *in-degree*. This also can be achieved by the corresponding operations on G. Actually, we first preprocess the adjacency lists of G before invoking the algorithm. One may note that this strategy is not applicable to dynamic graphs. In this paper, we focus on the unsolved static problem setting and leave the problem on dynamic graphs for future work.

Pruning Strategy. The practical efficiency of R^2LP still needs to be improved to handle large graphs as we empirically observe many node pairs with negligible similarities during the push procedure. Recall that R^2LP saves computational cost by adopting the randomized push strategy. We additionally propose a deterministic pruning strategy to directly rule out estimations with small values at each step l. To be precise, we set $f(c,\varepsilon) = c^l\varepsilon$ in Line 5 of Algorithm 3.

Correctness. The following lemma shows that our R^2LP algorithm guarantees additive error ε with high probability. Particularly, the above pruning strategy does not affect the error guarantee.



(a) Small graphs (b) Medium graphs

Figure 2: Distribution of all-pair SimRank values.

LEMMA 7. *By conducting the randomized push procedure of R^2LP for $O(\log\frac{n}{\delta})$ times and apply the Median-of-Mean trick [6], we have $|\hat{s}(u,v) - s(u,v)| \leq \varepsilon$ with at least $1-\delta$ probability.*

Complexity Analysis. We use the following theorem to guarantee the efficiency of R^2LP.

THEOREM 1. *The expected time complexity of R^2LP is bounded by $\tilde{O}(\frac{\sum_{u\in V}\sum_{v\in V}\sqrt{d_{in}(u)d_{in}(v)}s(u,v)}{\varepsilon})$.*

5 COMPUTATIONAL COMPLEXITY ANALYSIS

We analyze the computational complexity of threshold-based and top-k similarity joins with approximation bound ρ. By empirical study we observe the skewed SimRank distribution of real-world graphs. Meanwhile, existing literature [21, 34] has also demonstrated that single-source SimRank distribution approximately follows power law. This motivates us to make reasonable assumption of the distribution and then derive non-trivial complexity bounds that match the practical performance of algorithms.

5.1 Modeling SimRank Distribution

We demonstrate in Figure 2 the all-pair SimRank distributions of three small graphs and four medium graphs used in our experiments (Section 6). We omit the base cases with $s(v,v) = 1$. Generally the distributions show different degrees of skewness with a fraction of the top ranking node pairs having a similarity of c. Inspired by [2, 5], we propose the following assumption of SimRank distribution.

DEFINITION 5 (POWER LAW BOUNDED DISTRIBUTION OF SIMRANK). *Assume we have nnz node pairs with non-zero SimRank values for a given graph G. Let s_j denote the j-th largest SimRank value, for $j \in [1, nnz]$. We say the SimRank distribution is power law bounded (PLB) if for any $x = 1,\ldots,\lfloor\log_2 nnz\rfloor$, there exist parameters $\beta > 0$ and $b_2 > b_1 > 0$ such that*

$$\sum_{j=2^x}^{2^{x+1}-1} b_1 \cdot r(j)^{-\beta} \leq \sum_{j=2^x}^{2^{x+1}-1} s_j \leq \sum_{j=2^x}^{2^{x+1}-1} b_2 \cdot r(j)^{-\beta}, \quad (6)$$

where $r(j) = \max(1, j-t)$, and we set the shift $t = |\{(u,v)|s(u,v) = c, u,v \in G\}|$. Specifically, β is referred to as the PLB exponent of SimRank distribution, while b_1 and b_2 are small numbers that can be taken as constants.

Please refer to the full version of our paper [1] for empirical validation of the assumption.

5.2 Complexity of Threshold-based Queries

With the above assumption, we derive the *upper bound* of the complexity of threshold-based queries. We do this by bounding the complexity of our *APThres* (as well as *APTop-k*) framework which integrates an all-pair algorithm with absolute error guarantee.

LEMMA 8. *Given a graph G and an all-pair algorithm $\mathcal{AP}$ that guarantees absolute error, assume that APThres (or APTop-k) terminates when the error parameter reaches ε_t, and we have cost$(\mathcal{AP}(G, \varepsilon_{i+1})) \geq p \cdot$ cost$(\mathcal{AP}(G, \varepsilon_i))$ for $i \in [1, t)$ with $p > 1$. The time complexity of the algorithm is then bounded by $O($cost$(\mathcal{AP}(G, \varepsilon_t)))$.*

Remark. We can choose $\mathcal{AP}$ as *FLP*, *Opt-LP*, *KSimJoin*, or *R^{2}LP*. The lemma cannot be applied to *UISim* because it has no absolute error guarantee, nor *H-go SRJ* because the complexity depends on an algorithm-specific parameter h.

Next, we bound the error parameter ε_t for threshold-based queries with the help of Definition 5.

LEMMA 9. *We are given the problem input as in Lemma 8. Moreover, the graph G has PLB SimRank distribution. For threshold-based queries with approximation bound ρ, we have*

$$\varepsilon_t = O(\max(\frac{(b_1 - \rho^\beta b_2)\theta}{b_1 + \rho^\beta b_2}, \varepsilon_{min})). \quad (7)$$

The above lemma implies that given a graph G and an approximation bound ρ, the approximation guarantee can be non-trivially satisfied only if $b_1 - \rho^\beta b_2 > 0$, which is determined by both graph properties and the approximation parameter. The following theorem bounds the computational complexity of threshold-based queries.

THEOREM 2. *Given a graph G with PLB SimRank distribution, an approximation bound ρ with $b_1 - \rho^\beta b_2 > 0$, and an all-pair algorithm $\mathcal{AP}$ that guarantees absolute error, let cost$(\mathcal{AP}(G, \varepsilon))$ denote the complexity of $\mathcal{AP}$ given error parameter ε. The APThres framework with $\mathcal{AP}$ answers ρ-approximation threshold-based queries in $O($cost$(\mathcal{AP}(G, \frac{(b_1 - \rho^\beta b_2)\theta}{b_1 + \rho^\beta b_2})))$ time.*

5.3 Complexity of Top-k Queries

For top-k queries, the following theorem is provided, with the intuition that the complexity is mainly determined by the gap between the $\lceil \rho k \rceil$-th and the $(k+1)$-th largest values.

THEOREM 3. *We are given the problem input as in Lemma 9. For top-k queries with approximation bound ρ, with the PLB assumption of SimRank distribution, let $x_1 = 2^{\lceil \log_2 \lceil \rho k \rceil \rceil} - 1$ and $x_2 = 2^{\lfloor \log_2 (k+1) \rfloor}$. We have*

$$\varepsilon'_t = O(\max(b_1(x_1 - t)^{-\beta} - b_2(x_2 - t)^{-\beta}, \varepsilon_{min})). \quad (8)$$

Furthermore, with the power law assumption of all-pair SimRank distribution, i.e., $s_j = \Theta((j-t)^{-\beta})$, we have

$$\varepsilon'_t = O(\max((b_1(\lceil \rho k \rceil - t)^{-\beta} - b_2(k + 1 - t)^{-\beta}), \varepsilon_{min})). \quad (9)$$

The APTop-k framework with $\mathcal{AP}$ answers ρ-approximation top-k queries in $O($cost$(\mathcal{AP}(G, \varepsilon'_t)))$ time.

Lastly, we present a qualitative analysis of the complexities w.r.t. the input parameters for threshold-based and top-k queries. For both queries, the complexity upper bound increases as we set a higher approximation bound ρ. For threshold-based (resp. top-k) query, generally the problem tends to be harder as we decrease θ (resp. increase k). However, the PLB exponent β has different impacts on two types of queries. An illustrative demonstration is referred to Figure 3. For threshold-based queries, a smaller β (e.g. β_2) leads to a smaller ε_t in Lemma 9, and the complexity upper bound increases. Intuitively, more node pairs are included in the answer, while given an error parameter ε, the size of node pairs to be carefully handled (e.g., with similarity values between $\theta - \varepsilon$ and $\theta + \varepsilon$) also increases. In contrast, as Theorem 3 shows, a larger β (e.g. β_1) makes a smaller gap between the $\lceil \rho k \rceil$-th and the $(k+1)$-th SimRank values, so the algorithm needs more effort to separate them. This is in consistent with Figure 3, where the k-th SimRank value becomes smaller (also note that the figure is in log-log scale).

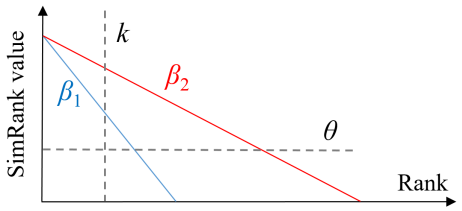


Figure 3: Illustration of the complexity of threshold-based and top-k similarity joins.

6 EXPERIMENTAL EVALUATION

We conduct experiments to evaluate the performance of state-of-the-art algorithms and our improvements. We focus on the following research questions:

RQ1: How do the algorithms perform empirically in answering threshold-based and top-k queries?

RQ2: Does the absolute error guarantee of an algorithm have influence on our problems?

RQ3: To what extent our proposed algorithms improve query efficiency or accuracy, and what are the unsolved problems?

RQ4: What are the impacts of the input parameters and graph properties on query efficiency?

6.1 Experimental Setup

Datasets. We use three small graphs (CG, WV, and EN), four medium graphs (SD, DB, WF, and ND), and three large graphs (CP, LJ, and WZ) for evaluation. We do not include larger datasets due to excessive memory consumption. All graphs are obtained from [3, 4, 14, 16] with their basic statistics listed in Table 3. In particular, $\bar{d}$ is the average degree and β is the fitted PLB exponent of all-pair SimRank distribution.

Ground truths. For small and medium graphs, we compute the all-pair ground truth using *Power Method* [11]. We set the number of iterations to 30 and achieve an absolute error about 10^{-7}. For medium graphs, we only store SimRank values above 10^{-3} due to excessive disk consumption. For large graphs, since *Power Method* is computationally intractable, we randomly choose 100 nodes in each graph, take each node as query input and use *ExactSim* [34] to compute the single-source ground truth with an absolute error of 10^{-7}. This partial-pair ground truth is then used for evaluation.

Table 3: Datasets and their statistics.

Dataset	Type	n	m	$\bar{d}$	β
ca-GrQc (CG)	U	5.2K	14.5K	2.77	0.67
Wiki-Vote (WV)	D	7.1K	103.7K	14.57	0.72
email-Enron (EN)	U	36.7K	183.8K	5.01	0.32
Slashdot0922 (SD)	D	82.2K	948.5K	11.54	0.38
DBLP (DB)	U	317.1K	1.05M	3.31	0.42
Wikilinks-fy (WF)	D	65.6K	1.07M	16.35	0.48
Notre Dame (ND)	D	325.7K	1.5M	4.6	0.27
cit-Patents (CP)	D	3.77M	16.52M	4.38	-
LiveJournal (LJ)	D	4.85M	68.99M	14.23	-
Wikilinks-zh (WZ)	D	1.79M	72.61M	40.65	-

Baselines. We include *FLP & Opt-LP* [37], *UISim* [45], and *K-SimJoin* [19] as our baselines. We do not include [24, 26, 27, 30] as they are outperformed by the baselines according to their empirical evaluation. The implementation of [37, 45] is obtained from the authors. We implement other baselines and our improved algorithms in C++. Specifically, we do not consider *H-go SRJ* [43] for efficiency issues. We find that it incurs tremendous time and space cost for index construction and query execution even on small graphs, with significant inferior performance compared to other baselines.

We conduct all experiments on an Ubuntu server with an Hygon C86 7151 processor and 1TB memory, repeat each experimental settings for ten times and report the average measures.

6.2 Evaluation of Additive Error

We first evaluate the additive error of each algorithm in answering general all-pair queries, since they are the building blocks of threshold-based and top-k algorithms.

Experimental settings. For algorithms with absolute error guarantee, including *FLP*, *Opt-LP*, *R^{2}LP* and *KSimJoin*, we vary the error parameter ε. For small graphs (CG, WV, and EN) and medium graphs (SD, DB, WF, and ND), we set $\varepsilon = \{0.05, 0.01, 0.005, 0.001\}$. For large graphs (CP, LJ, and WZ), we vary $\varepsilon = \{0.1, 0.05, 0.01\}$. In particular, we slightly modify *KSimJoin* to answer all-pair queries by excluding the upper bound computation and the iterative pruning framework. For *UISim*, we fix the number of iterations $\eta = 2$ and vary the number of hub nodes $|H| = \{0.2n, 0.5n\}$. As for its implementation, the authors adopt another parameter named stopping reachability ($stopRea$), where the graph expansion stops either when a hub node is encountered or the walking probability falls below $stopRea$. As we empirically find that $stopRea$ has more impact on query efficiency and accuracy, we vary it in $\{0.005, 0.001\}$ for small graphs and $\{0.05, 0.01\}$ for medium graphs, and set it as 0.01 for large graphs. We fix $L = 2$ for *KSimJoin*.

Evaluation metric. We use $AvgErr$ for evaluation, which is defined as $AvgErr = \frac{\sum_{(u,v)\in S(\theta)} |s^*(u,v) - \hat{s}(u,v)|}{|S(\theta)|}$, where $s^*(u,v)$ and $\hat{s}(u,v)$ are the ground truth and estimated value respectively, and $S(\theta) = \{(u,v)|s^*(u,v) \geq \theta, u, v \in V, u \neq v\}$. We set $\theta = 10^{-3}$ for the following reasons. Applications usually focus on non-negligible SimRank values, which only make up a small portion of the SimRank matrix. With most other similarity values approaching zero, it is not wise to take them into consideration, as an algorithm that simply returns zero for each node pair achieves very small average error. Moreover, we cannot afford to compute and store the ground truths for node pairs with very small similarities for medium and large graphs.

The results are demonstrated in Figure 4. Throughout the paper, all reported query times are in seconds. For *UISim*, the unfilled markers corresponds to the default values of $stopRea$ (0.001 and 0.01), while the filled markers refer to the increase of $stopRea$ by 5 times (i.e., 0.005 and 0.05) on small and medium graphs. Surprisingly, although *UISim* does not guarantee the absolute error, it outperforms *FLP* and *Opt-LP* on 6 of 10 graphs, and achieves comparable performance on another graph (DB). However, we note that varying the number of hubs has quite limited influence on both query efficiency or accuracy, that is, the two unfilled markers (as well as the two filled markers) are very close, especially for WV, EN, SD, WF, and LJ. By contrast, the filled and unfilled markers show very different performance. The reason is that most of graph expansions terminate because of stopping reachability rather than encountering a hub node. As we increase $stopRea$, the algorithm returns SimRank estimations of larger additive errors with less response time.

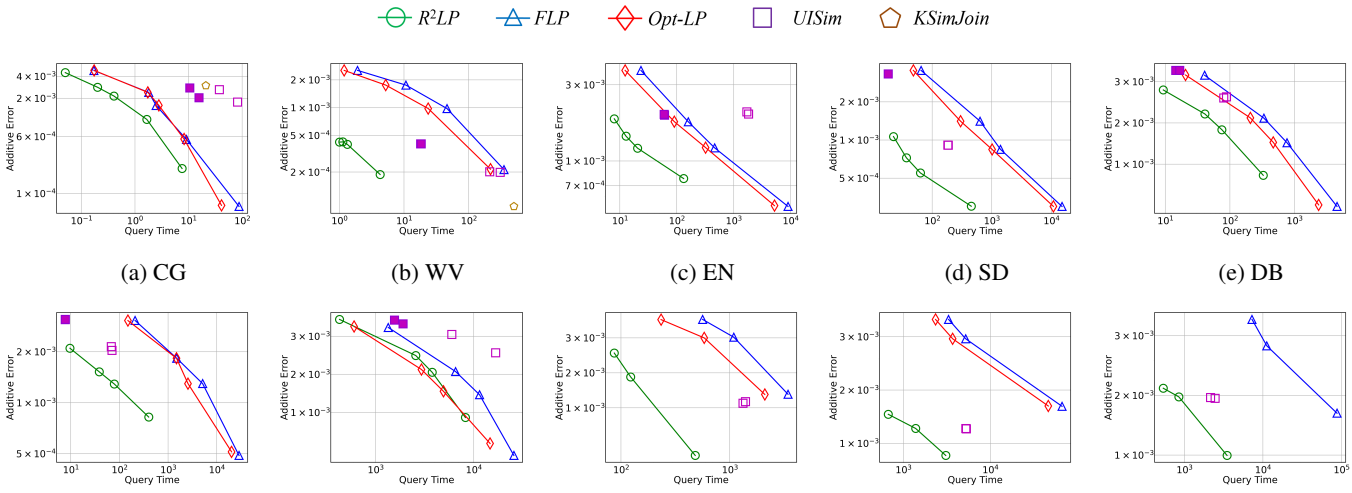
For algorithms with absolute error guarantees, *Opt-LP* outperforms *FLP* on most graphs thanks to the optimizations, except that it fails on the largest graph WZ (not terminated for over 10 hours). However, they are usually slower than *UISim*, because backward push on the SimRank graph is very time consuming. *KSimJoin* only manages to answer all-pair queries on the two smallest graphs and has similar performance as *UISim*, while the tremendous memory consumption makes it infeasible on larger graphs. For example, on WV and with $L = 2$, the NP structure for all nodes has about 1.8M items. On the contrary, with the randomized push strategy, *R^{2}LP* achieves significantly better performance compared to all other baselines. On most graphs, it is faster than *FLP*, *Opt-LP*, and *KSimJoin* by up to an order of magnitude with similar additive errors. The only exception is ND, where *R^{2}LP* and *Opt-LP* show comparable results. Recall that ND has numerous node pairs with very large similarity values (see Figure 2(b)), thus the randomized part of *R^{2}LP* is seldomly invoked. In general, to achieve good results, we need both theoretical guarantee of the algorithm as well as strategies to ensure practical efficiency.

6.3 Evaluation of Threshold-based Queries

Experimental settings and metrics. For small and medium graphs, we directly evaluate the results of all-pair threshold-based queries. For three large graphs, as we only have the ground truth of 100 single-source queries, we ignore the algorithm outputs that are not contained in this ground truth. We adopt *Precision*, *Recall* and *F1-score* for evaluation. Given a threshold-based query with parameter $\theta \in (0, c]$, let $R^*(\theta)$ denote the set of node pairs with ground truth SimRank values above or equal to θ (with up to $\varepsilon_{min} = 10^{-7}$ error), and we assume $|R^*(\theta)| > 0$. Let $R_{\mathcal{A}}(\theta)$ denote the returned set by algorithm $\mathcal{A}$. We have *Precision* $= |R_{\mathcal{A}}(\theta) \cap R^*(\theta)|/|R_{\mathcal{A}}(\theta)|$, *Recall* $= |R_{\mathcal{A}}(\theta) \cap R^*(\theta)|/|R^*(\theta)|$, and *F1-score* $= 2/(\frac{1}{\textit{Precision}} + \frac{1}{\textit{Recall}})$.

In our first set of experiments, we vary $\theta = \{0.1, 0.01, 0.001\}$ and evaluate the all-pair algorithms used in Section 6.2. We only include three best algorithms *R^{2}LP*, *Opt-LP*, and *UISim*. Note that none of these algorithms provides approximation bound for threshold-based queries, but we can still compare their empirical performance. The results are demonstrated in Figure 5[3] with the

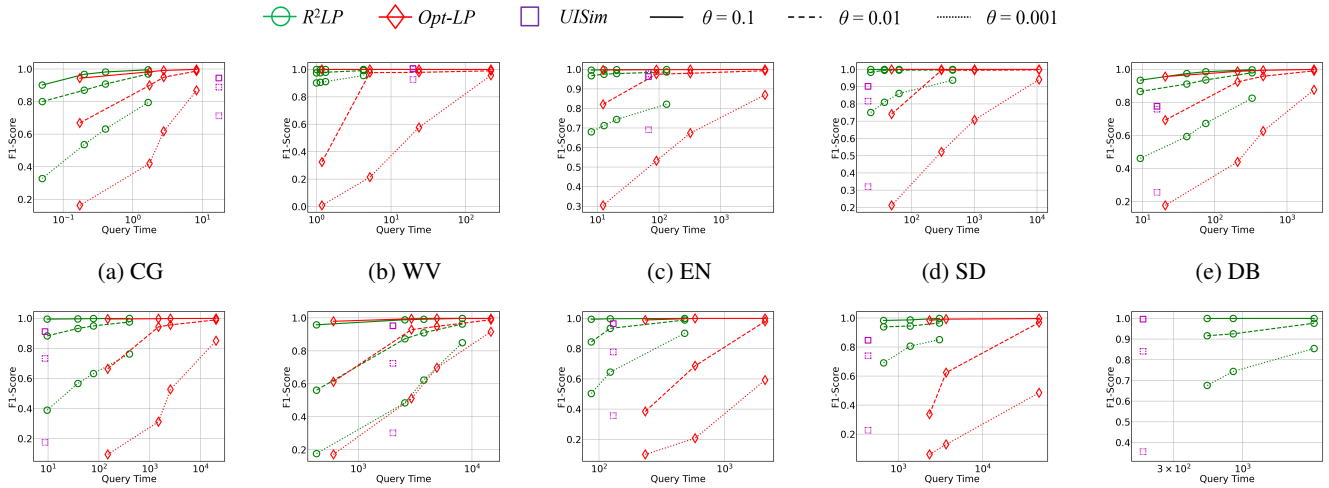
[3]For the results of *Precision* and *Recall*, please refer to the full version of our paper [1].



(a) CG (b) WV (c) EN (d) SD (e) DB

(f) WF (g) ND (h) CP (i) LJ (j) WZ

Figure 4: Additive error of SimRank estimation.



(a) CG (b) WV (c) EN (d) SD (e) DB

(f) WF (g) ND (h) CP (i) LJ (j) WZ

Figure 5: F1-score for threshold-based queries. No approximation bound is provided.

following conclusions derived. First, for each algorithm and each of its parameter configuration, the *F1-scores* vary significantly for different graphs and with different θ. It is not recommended to answer the threshold-based query with a general all-pair algorithm by simply setting $\varepsilon = O(\theta)$. For example, on CG, EN, DB, and WF, the *F1-scores* with $\varepsilon = \theta = 0.001$ are below 0.9. Second, R^2LP generally achieves the best tradeoff between efficiency and accuracy except on ND. Third, for *UISim*, we choose the best one among the four configurations used in Section 6.2. On a majority of graphs, it surpasses *Opt-LP* mainly due to the fast query time. However, as we decrease $stopRea$ and increase η, the algorithm fails to compute an output in reasonable time (say 3 hours), which implies the hub-based strategy is less flexible than the probability-guided ones.

Next, we apply the *APThres* framework and integrate it with R^2LP and *Opt-LP*, the best algorithms with absolute error guarantee. By fixing $\theta = 0.01$, we test their actual performance with different approximation bounds, which is shown in Table 4. Consistent with our theoretical guarantee, both *APThres+Opt-LP* and *APThres+*R^2LP hold the approximation bound on all graphs and all settings of ρ. *APThres+*R^2LP achieves similar *F1-score* with much less query time except on ND, of which the reasons are explained in Section 6.2. Also note that the advantage of *Opt-LP* over R^2LP with an iterative framework lies in that it can compute from the "borders" of the previous iteration rather than from scratch.

However, compared to Figure 5, both algorithms incur significantly more time, and fail on LJ for different reasons[4] while *APThres+Opt-LP* also fails on WZ. Recall that R^2LP is able to handle LJ with acceptable accuracy if we do not force an approximation bound.

[4]We say an algorithm is time out if it takes over an hour in one iteration of *APThres*.

Table 4: Threshold-based queries with $\theta = 0.01$. We vary $(\rho_1, \rho_2) = (0.9, 0.99), (0.9, 0.95), (0.7, 0.9)$ for small, medium and large graphs respectively.

	Opt-LP (ρ_1)		*Opt-LP* (ρ_2)		*R²LP* (ρ_1)		*R²LP* (ρ_2)	
	Time	F1	Time	F1	Time	F1	Time	F1
CG	5.55	0.991	11.99	0.997	2.85	0.978	8.28	0.993
WV	152.4	0.998	159.7	0.998	31.2	0.998	32.0	0.999
EN	2944	0.995	10625	0.999	201.8	0.989	1040	0.996
SD	9451	0.998	12011	0.998	611.6	0.997	1457	0.998
DB	1994	0.994	3480	0.997	501.6	0.985	1119	0.992
WF	13336	0.994	8938	0.994	649.9	0.984	1370	0.99
ND	10691	0.991	13129	0.996	6607	0.936	10269	0.958
CP	5487	0.999	8504	0.999	1903	0.999	4350	0.999
LJ	Time Out				Out of Memory (OOM)			
WZ	Time Out				3743	0.985	7411	0.99

Meanwhile, the actual *F1-scores* shown in Table 4 far exceed the bound ρ. We believe this phenomenon is attributed to the pessimistic estimation of the absolute error, that is, for almost all node pairs, it turns out that $|\hat{s}(u,v) - s(u,v)| \ll \varepsilon$. We have tried other optimization strategies (see Section 4.1) and find that for now *R²LP* is the most efficient algorithm to guarantee an absolute error. Hence, it remains an open problem to answer threshold-based queries more efficiently and with theoretical approximation bound.

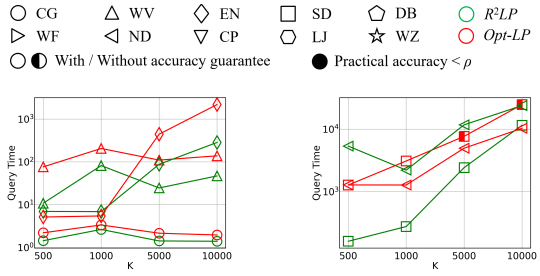
6.4 Evaluation of Top-k Queries

Experimental settings and metrics. For top-k similarity joins, *Precision@k* is used for evaluation. Let $R^*(k)$ and $R_{\mathcal{A}}(k)$ denote the ground truth and the results of algorithm $\mathcal{A}$, respectively. We have $\textit{Precision@}k = |R_{\mathcal{A}}(k) \cap R^*(k)|/k$. As shown in Figure 2, numerous node pairs have similarity value c, and the numbers of such node pairs vary significantly for different graphs. Instead of setting different k for our datasets, the following strategy is adopted. For medium and large graphs, we conduct *all-pair queries* but only evaluate the result accuracy on a pair of query node sets A and B, which is referred to as *partial-pair evaluation*. Specifically, we set $A = B = V$ for CG and WV, while for EN, SD, WF, and ND we set $|A| = |B| = 10^3$. Since the all-pair ground truth of DB is very sparse, we set $|A| = |B| = 10^4$. For three large graphs, we fix A as the 100 query nodes in generating ground truth, and set $|B| = 10^3$. We randomly sample the query node sets and repeat the experiment for ten times. Note that we only use partial-pair evaluation to facilitate the comparison of different algorithms and do not consider the possible optimizations for partial-pair queries.

We first compare the state-of-the-art algorithms by fixing $k = 5000$ and $\rho = 0.9$ with the results listed in Table 5. For *UISim*, we fix $\eta = 2$ and set $stopRea$ as 10^{-4} for CG and WV, 10^{-3} for EN, and 10^{-2} for all other datasets. For *KSimJoin*, it shares the same parameter settings as in Section 6.2. We integrate *Opt-LP* and *R²LP* with *APTop-k* to answer top-k queries. Particularly, we terminate the algorithm early if it takes more than an hour in one iteration, and set it as time out if it costs over 30000 seconds. Although being fast, *UISim* mainly suffers from accuracy issues. On the other hand, *KSimJoin* has severe scalability problem on medium and large graphs in spite of its pruning strategy. For both *APTop-k+Opt-LP* and *APTop-k+R²LP*, the query accuracy is guaranteed.

Table 5: Precision@5000 and query time ($\rho = 0.9$). The practical accuracy is underlined if it falls below ρ.

	UISim		*Opt-LP*		*KSimJoin*		*R²LP*	
	Time	P@k	Time	P@k	Time	P@k	Time	P@k
CG	82.5	0.935	2.16	0.999	6.3	0.926	1.42	0.998
WV	308.2	0.992	110.3	0.999	405.9	0.994	24.6	0.998
EN	1189	0.993	443.7	0.997	OOM		88.0	0.996
SD	185.4	0.922	7682	0.996	OOM		2424	0.997
DB	84.0	0.777	1082	0.998	OOM		333.7	0.995
WF	56.0	0.776	10501	0.99	OOM		3488	0.995
ND	13448	0.959	4964	0.997	OOM		11749	0.993
CP	1416	0.95	2037	0.992	OOM		1190	0.994
LJ	5604	0.917	8586	0.964	OOM		8824	0.97
WZ	2375	0.904	Time Out		OOM		7171	1



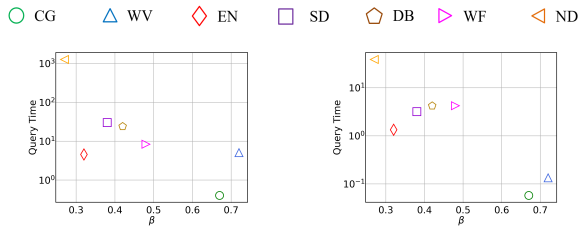
(a) CG, WV, EN (b) SD, ND

(c) DB, WF (d) CP, LJ, WZ

Figure 6: Query time vs. k, with $\rho = 0.9$.

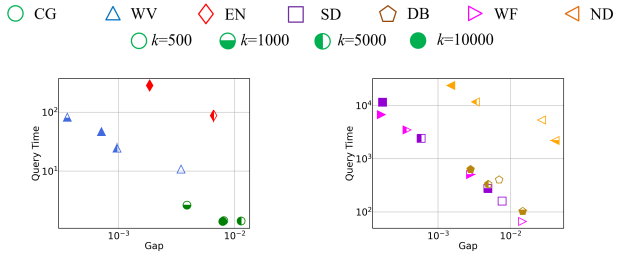
With theoretical bound, they also face the open problem as for the threshold-based queries. Nonetheless, *APTop-k+R²LP* is the fastest on 5 of 10 graphs while achieving much better practical accuracy than its counterparts.

Next, we evaluate the query efficiency w.r.t. k by setting $k = \{500, 1000, 5000, 10000\}$. Only *APTop-k+Opt-LP* and *APTop-k+R²LP* are considered because other methods fail to guarantee stable accuracy. As demonstrated in Figure 6, the red (resp. green) line denotes the result of *Opt-LP* (resp. *R²LP*), while different types of markers indicate whether the result has theoretical guarantee or even has inferior accuracy than the bound. Recall that if the algorithm terminates earlier due to excessive time cost, it is unable to guarantee the theoretical accuracy. In most cases, *APTop-k+R²LP* outperforms *APTop-k+Opt-LP* by a significant margin. On three large graphs CP, LJ, and WZ, both algorithms lose the theoretical guarantee due to early termination. Fortunately, the practical accuracy of *APTop-k+R²LP* is always above ρ, whereas *APTop-k+Opt-LP* has inferior accuracy on LJ with $k = 10000$ and even fails on WZ.



(a) Threshold queries (b) Top-k queries

Figure 7: Query time vs. β.



(a) Small graphs (b) Medium Graphs

Figure 8: Query time vs. $s_{\lceil \rho k \rceil} - s_{k+1}$ for top-k queries.

Another interesting observation from Figure 6 is that the query time does not necessarily increase when we enlarge k. Generally, both algorithms demonstrate similar trends on all graphs. The explanation is given in the next subsection.

6.5 Empirical Study of Computational Complexity

We discuss the impact of approximation bound ρ and input parameters θ and k on query efficiency, which can be concluded from the experiments in Section 6.3 & 6.4. For example, Figure 5 and Table 4 imply that a larger approximation bound ρ or a smaller parameter θ leads to more time cost, while Figure 6 directly suggests the time complexity of top-k queries generally increases as we enlarge k. For a deeper investigation of the algorithmic complexities, we further conduct two sets of experiments to validate our theoretical analysis w.r.t. graph properties. As *R^2LP* and *Opt-LP* demonstrate similar performance in the above experiments (except for *R^2LP* being faster), we only use *R^2LP* in the following analysis.

Firstly, we conduct threshold queries with $\theta = 0.01$ and top-k queries with $k = 1000$ following the settings in the above subsections. To illustrate the intrinsic problem complexities, we make sure the *practical accuracy* of *R^2LP* slightly surpasses $\rho = 0.9$ by adjusting the error parameter ε and plot in Figure 7 the query time w.r.t. β, the PLB exponent of SimRank distribution. Note that the graph size do have an important impact on query efficiency, which is consistent with the computational complexity of *R^2LP* (Theorem 1). However, for graphs with similar sizes, such as EN, SD, DB and WF, the query time is in general negatively (resp. positively) correlated with β for threshold-based (resp. top-k) queries, which coincides with the derived computational complexity (Theorem 2 & 3).

Secondly, for top-k queries, our theoretical analysis suggests that the time complexity heavily relies on the gap between the $\lceil \rho k \rceil$-th and the $(k+1)$-th largest SimRank values. Figure 8 directly shows query time w.r.t. the gap for small and medium graphs, where the all-pair ground truths are available. It turns out that query time nicely correlates with the gap on all datasets. Note that for a majority of studied graphs including CG, WV, DB and ND, the gap does not necessarily decrease as we increase k. This explains the phenomenon observed in Figure 6. It is worthy pointing out that the observed correlation also fits the sampling complexity of the top-k identification problem [9, 13] in the multi-armed bandit setting, where the backward push procedure (such as *R^2LP* and *Opt-LP*) is considered as generating an estimation for each arm (i.e., node pair) with ε confidence interval.

7 OTHER RELATED WORK

To accelerate the *Power Method*, early work [22, 42] improves the algorithmic complexity by pruning redundant or unnecessary computation. Another line of work [10, 17, 39] studies all-pair SimRank computation on dynamic graphs. We also note that several all-pair algorithms [8, 38–41] rely on a modified definition of SimRank, i.e. $S = cP^\intercal SP + (1-c)I$, which results in estimations without absolute error guarantee.

As for other types of SimRank queries, most of the existing work (e.g., [12, 15, 20, 21, 28, 29, 31, 34]) focuses on single-source settings. Given a query node u, the returned result can either be the estimated SimRank value w.r.t. u for *each* node v [12, 28, 29, 31, 34], the set of nodes with top-k largest SimRank values (the *single-source top-k query* [15, 20, 21]), or all nodes whose similarity w.r.t. u are above some threshold θ (the *single-source thresholding query* [21]). Various algorithms have been proposed based on random walk sampling [12, 28], graph traversal [15, 20, 21, 31], and local push [29], while recent literature [20, 21, 29, 34] has significantly improved the query efficiency on large graphs. For example, SimPush [29] answers single-source queries on billion-edge graphs in seconds with additive error less than 10^{-3}. Besides, a single-pair query takes two query nodes u and v as input and computes an estimation of $s(u,v)$ with additive error up to ε. Existing methods such as the Monte Carlo (*MC*) algorithm [7] and its recent improvement [36] can answer single-pair queries within milliseconds for $\varepsilon \geq 10^{-4}$.

8 CONCLUSION

This paper presents detailed empirical studies, new algorithms with better efficiency and approximation bounds, and novel theoretical analysis for two types of all-pair SimRank queries, namely, threshold-based and top-k similarity joins. We conduct a detailed comparison of state-of-the-art algorithms followed by the introduction of algorithm frameworks that theoretically guarantee the result accuracy for both queries, along with a more efficient all-pair algorithm inspired by the randomized local push. To gain a deep understanding of algorithm performance and the computational complexities of these problems, we propose theoretical analysis by leveraging the SimRank distribution and conduct extensive experimental studies. Our experiments show that the proposed algorithms significantly improve both query efficiency and accuracy, while our theoretical results nicely match the empirical analysis. For future work, we aim to study all-pair SimRank computation in dynamic settings.

REFERENCES

[1] [n.d.]. https://github.com/xinghun0525/R2LP.

[2] Jonathan W Berry, Luke K Fostvedt, Daniel J Nordman, Cynthia A Phillips, C Seshadhri, and Alyson G Wilson. 2014. Why do simple algorithms for triangle enumeration work in the real world?. In *Proceedings of the 5th conference on Innovations in theoretical computer science*. 225–234.

[3] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. 2011. Layered Label Propagation: A MultiResolution Coordinate-Free Ordering for Compressing Social Networks. In *Proceedings of the 20th international conference on World Wide Web*, Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar (Eds.). ACM Press, 587–596.

[4] Paolo Boldi and Sebastiano Vigna. 2004. The WebGraph Framework I: Compression Techniques. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*. ACM Press, Manhattan, USA, 595–601.

[5] Pawel Brach, Marek Cygan, Jakub Lkacki, and Piotr Sankowski. 2016. Algorithmic complexity of power law networks. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1306–1325.

[6] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. 2002. Finding Frequent Items in Data Streams. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings (Lecture Notes in Computer Science)*, Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan J. Eidenbenz, and Ricardo Conejo (Eds.), Vol. 2380. Springer, 693–703. https://doi.org/10.1007/3-540-45465-9_59

[7] Dániel Fogaras and Balázs Rácz. 2005. Scaling link-based similarity search. In *Proceedings of the 14th international conference on World Wide Web*. 641–650.

[8] Yasuhiro Fujiwara, Makoto Nakatsuji, Hiroaki Shiokawa, and Makoto Onizuka. 2013. Efficient search algorithm for SimRank. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 589–600.

[9] Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. 2012. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems*. 3212–3220.

[10] Guoming He, Haijun Feng, Cuiping Li, and Hong Chen. 2010. Parallel SimRank computation on large graphs with iterative aggregation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 543–552.

[11] Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *SIGKDD*. 538–543.

[12] Minhao Jiang, Ada Wai-Chee Fu, and Raymond Chi-Wing Wong. 2017. READS: a random walk approach for efficient and accurate dynamic SimRank. *PVLDB* 10, 9 (2017), 937–948.

[13] Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. 2012. PAC Subset Selection in Stochastic Multi-armed Bandits.. In *ICML*, Vol. 12. 655–662.

[14] Jérôme Kunegis. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd international conference on world wide web*. 1343–1350.

[15] Pei Lee, Laks VS Lakshmanan, and Jeffrey Xu Yu. 2012. On top-k structural similarity search. In *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 774–785.

[16] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[17] Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu. 2010. Fast computation of simrank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology*. 465–476.

[18] Lina Li, Cuiping Li, Hong Chen, and Xiaoyong Du. 2013. Mapreduce-based SimRank computation and its application in social recommender system. In *2013 IEEE international congress on big data*. IEEE, 133–140.

[19] Ruiqi Li, Xiang Zhao, Haichuan Shang, Yifan Chen, and Weidong Xiao. 2017. Fast top-k similarity join for SimRank. *Inf. Sci.* 381 (2017), 1–19. https://doi.org/10.1016/j.ins.2016.10.042

[20] Yu Liu, Bolong Zheng, Xiaodong He, Zhewei Wei, Xiaokui Xiao, Kai Zheng, and Jiaheng Lu. 2017. Probesim: scalable single-source and top-k simrank computations on dynamic graphs. *arXiv preprint arXiv:1709.06955* (2017).

[21] Yu Liu, Lei Zou, Qian Ge, and Zhewei Wei. 2020. SimTab: accuracy-guaranteed simrank queries through tighter confidence bounds and multi-armed bandits. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2202–2214.

[22] Dmitry Lizorkin, Pavel Velikhov, Maxim Grinev, and Denis Turdakov. 2008. Accuracy Estimate and Optimization Techniques for SimRank Computation. *Proc. VLDB Endow.* 1, 1 (aug 2008), 422C433. https://doi.org/10.14778/1453856.1453904

[23] Peter Lofgren and Ashish Goel. 2013. Personalized pagerank to a target node. *arXiv preprint arXiv:1304.4658* (2013).

[24] Juan Lu, Zhiguo Gong, and Xuemin Lin. 2016. A novel and fast simrank algorithm. *IEEE transactions on knowledge and data engineering* 29, 3 (2016), 572–585.

[25] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390, 6 (2011), 1150–1170.

[26] Takanori Maehara, Mitsuru Kusumoto, and Ken-ichi Kawarabayashi. 2014. Efficient SimRank computation via linearization. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 1426–1435. https://doi.org/10.1145/2623330.2623696

[27] Takanori Maehara, Mitsuru Kusumoto, and Ken-ichi Kawarabayashi. 2015. Scalable SimRank join algorithm. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman (Eds.). IEEE Computer Society, 603–614. https://doi.org/10.1109/ICDE.2015.7113318

[28] Yingxia Shao, Bin Cui, Lei Chen, Mingming Liu, and Xing Xie. 2015. An efficient similarity search framework for SimRank over large dynamic graphs. *PVLDB* 8, 8 (2015), 838–849.

[29] Jieming Shi, Tianyuan Jin, Renchi Yang, Xiaokui Xiao, and Yin Yang. 2020. Realtime Index-Free Single Source SimRank Processing on Web-Scale Graphs. *Proc. VLDB Endow.* 13, 7 (2020), 966–978. https://doi.org/10.14778/3384345.3384347

[30] Wenbo Tao, Minghe Yu, and Guoliang Li. 2014. Efficient Top-K SimRank-based Similarity Join. *Proc. VLDB Endow.* 8, 3 (2014), 317–328. https://doi.org/10.14778/2735508.2735520

[31] Boyu Tian and Xiaokui Xiao. 2016. SLING: A Near-Optimal Index Structure for SimRank. In *SIGMOD*. 1859–1874.

[32] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 world wide web conference*. 539–548.

[33] Hanzhi Wang, Zhewei Wei, Junhao Gan, Sibo Wang, and Zengfeng Huang. 2020. Personalized pagerank to a target node, revisited. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 657–667.

[34] Hanzhi Wang, Zhewei Wei, Yu Liu, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2021. ExactSim: benchmarking single-source SimRank algorithms with high-precision ground truths. *The VLDB Journal* 30, 6 (2021), 989–1015.

[35] Sibo Wang, Renchi Yang, Xiaokui Xiao, Zhewei Wei, and Yin Yang. 2017. FORA: simple and effective approximate single-source personalized pagerank. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 505–514.

[36] Yue Wang, Lei Chen, Yulin Che, and Qiong Luo. 2019. Accelerating pairwise simrank estimation over static and dynamic graphs. *The VLDB Journal* 28, 1 (2019), 99–122.

[37] Yue Wang, Xiang Lian, and Lei Chen. 2018. Efficient simrank tracking in dynamic graphs. In *2018 IEEE 34th international conference on data engineering (ICDE)*. IEEE, 545–556.

[38] Weiren Yu, Xuemin Lin, and Wenjie Zhang. 2013. Towards efficient SimRank computation on large networks. In *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE, 601–612.

[39] Weiren Yu, Xuemin Lin, and Wenjie Zhang. 2014. Fast incremental simrank on link-evolving graphs. In *2014 ieee 30th international conference on data engineering*. IEEE, 304–315.

[40] Weiren Yu, Xuemin Lin, Wenjie Zhang, Lijun Chang, and Jian Pei. 2013. More is simpler: Effectively and efficiently assessing node-pair similarities based on hyperlinks. *Proceedings of the VLDB Endowment* 7, 1 (2013), 13–24.

[41] Weiren Yu and Julie A McCann. 2015. Efficient partial-pairs simrank search on large networks. *Proceedings of the VLDB Endowment* 8, 5 (2015), 569–580.

[42] Weiren Yu, Wenjie Zhang, Xuemin Lin, Qing Zhang, and Jiajin Le. 2012. A space and time efficient algorithm for SimRank computation. *World Wide Web* 15, 3 (2012), 327–353.

[43] Weiguo Zheng, Lei Zou, Lei Chen, and Dongyan Zhao. 2017. Efficient SimRank-Based Similarity Join. *ACM Trans. Database Syst.* 42, 3 (2017), 16:1–16:37. https://doi.org/10.1145/3083899

[44] Weiguo Zheng, Lei Zou, Yansong Feng, Lei Chen, and Dongyan Zhao. 2013. Efficient SimRank-based Similarity Join Over Large Graphs. *Proc. VLDB Endow.* 6, 7 (2013), 493–504. https://doi.org/10.14778/2536349.2536350

[45] Fanwei Zhu, Yuan Fang, Kai Zhang, Kevin Chang, Hongtai Cao, Zhen Jiang, and Minghui Wu. 2021. Unified and Incremental SimRank: Index-free Approximation with Scheduled Principle. *IEEE Transactions on Knowledge and Data Engineering* (2021).