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 [14] is one of the most important measures for pairwise node similarity, which finds its applications in recommender systems [22], link prediction [30], and graph embeddings [37]. 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":

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

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 [36, 39, 42] or 0.8 [14]. Due to its computational complexity, efficient computation of SimRank has been extensively studied in the past two decades [9, 19, 21, 25, 27, 29, 31, 35], and remains to be a hot topic in the very recent years [26, 34, 39, 42, 48, 52].

According to the query inputs and outputs, SimRank queries can be broadly categorized as *single-pair* queries, *single-source* queries, and *all-pair* queries with the following definition.

DEFINITION 1 (ALL-PAIR SIMRANK COMPUTATION). *Given a graph $G = (V, E)$ of n nodes and an error parameter ε, return the SimRank estimation $\hat{s}(u,v)$ for every node pair (u,v) with absolute error guarantee, so that $|\hat{s}(u,v) - s(u,v)| \leq \varepsilon$ (with high probability).*

Among existing work [23, 29, 31, 32, 35, 42, 48, 50–52] that tackles all-pair queries, state-of-the-art algorithms [42, 52] can hardly guarantee an additive error of $\varepsilon = 0.01$ on million-node graphs. Besides, a recent work [39] indicates that it is "essentially hopeless" to compute all-pair queries on large graphs, as non-zero items can be as large as $O(n^2)$. Fortunately, two modified versions of all-pair queries have been studied, namely, *threshold-based* [32, 50, 51] and *top-k* [23, 35, 52] *similarity joins*, of which the outputs are small subsets of all-pair SimRank computation. Both queries are more practically useful because we are only interested in the node pairs with non-negligible similarities [52].

DEFINITION 2 (THRESHOLD-BASED SIMILARITY JOIN). *Given a graph $G = (V, E)$ and a threshold $\theta > 0$, return the set of node pairs $\mathcal{R}(\theta)$ with $u \neq v$ and SimRank value $s(u, v) \geq \theta$.*

DEFINITION 3 (TOP-k SIMILARITY JOIN). *Given a graph $G = (V, E)$ and a parameter $k > 0$, return a k-sized set $\mathcal{R}(k)$ containing the node pairs with the top-k largest SimRank values among all pairs $\{(u, v)|u, v \in V, u \neq v\}$.*

Analogous to all-pair SimRank computation that an error parameter ε is allowed, we introduce the *approximation bound* ρ for similarity joins. As we shall see in Section 4, without ρ (i.e., setting $\rho = 1$), in worst cases no algorithm can be more efficient than computing the exact SimRank values[1].

DEFINITION 4 (APPROXIMATION BOUND). *Given an approximation parameter $\rho \in (0, 1)$, we say an algorithm $\mathcal{A}$ has approximation bound 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 [42, 52] of general all-pair queries, the result accuracy is not evaluated except for small graphs due to lack of ground truth. Other representative algorithms [23, 50] 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. As far as we know, it is unclear how the input parameters and graph properties impact the algorithmic complexity of both queries.

1.2 Contributions

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

Detailed analysis of state-of-the-art algorithms. We choose four representative algorithms, *UISim* [52], *FLP & Opt-LP* [42], *H-go SRJ* [50], and *KSimJoin* [23] 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 [52]. *H-go SRJ* and *KSimJoin* are the state-of-the-art algorithms for threshold-based and top-k similarity joins, respectively. Apart from their performance, we choose *UISim* [52] and *KSimJoin* [23] because they adopt the random walk interpretation, whereas *Opt-LP* [42] and *H-go SRJ* [50] are the best methods based on the SimRank matrix formation. 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) 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 [42], and devise Randomized Reverse Local Push (R^2LP) inspired by randomized backward push for PPR [38]. It improves the all-pair query complexity from $O(\frac{\sum_{u,v \in V} d_{in}(u) d_{in}(v) s(u,v)}{\varepsilon})$ (the best known algorithm [42]) 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.

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. Our empirical analysis validates that for most real-world graphs, the ground truth SimRank values follow some *generalized* version of power-law distribution [2, 6], which is consistent with previous studies [26, 39]. 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 to 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 [14]. 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 [47], i.e., $S = cP^\intercal SP \vee I$, where S is an $n \times n$-dimensional similarity matrix with $S[u, v] = s(u, v)$, 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* [14], which is essentially a fixed-point iteration process on G^2. Recent

[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 a very accurate approximation.

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 [14] and the SimRank graph [42]
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

literature [21, 29, 42] 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 [14]. We adopt $\sqrt{c}$-walk, its revised version following [15, 25, 36]. Let $W_{\sqrt{c}}(u)$ denote a random walk starting from node u and following incoming edges at each step with walking probability $\sqrt{c}$. At each step, with probability $\sqrt{c}$, it randomly chooses an in-neighbor and continues the walk, otherwise the walk stops at the current node. [36] 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 (see Figure 1 and Table 2). We choose four methods including *UISim* [52], *FLP & Opt-LP* [42], *H-go SRJ* [50] and *KSimJoin* [23] for their state-of-the-art performance. *UISim* [52] and *KSimJoin* [23] are the latest methods based on the random walk interpretation, whereas *FLP & Opt-LP* [42] and *H-go SRJ* [50] are the best algorithms relying on the SimRank matrix formation. We also correct the erroneous claims and make up the missing theoretical analysis in the original papers.

3.1 *UISim*

The algorithm depends on the random walk interpretation [14] which says that SimRank value $s(u, v)$ equals to the summation of *first-meeting probabilities* of all random walk pairs starting from u and v, respectively. To compute $s(u, v)$, *UISim* adopts the following idea: *"Approximately prioritize all random walk pairs and first handle those with high meeting probabilities."* In practice, this is done by selecting a set of hub nodes H with high *in-degrees*. Note that a random walk passing many hubs will have small walking probability. Therefore, *UISim* prioritizes a random walk pair by the number of encountered hubs, which is an approximation of ordering walk pairs by their meeting probabilities.

For all-pair SimRank computation, *UISim* first computes the *prime out-subgraphs* of all nodes with at least two out-neighbors (denoted as $V^o_{\geq 2}$), and then assembles all generated random walk pairs (level-1 expansion in Figure 1(a)). The prime out-subgraph of node v contains all reachable nodes starting from v and following out-edges until a hub node is met. Next, we expand all encountered hubs with their prime out-subgraphs (level-2 expansion). The algorithm terminates with level-η expansion.

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

Complexity Analysis. According to [52], the time complexity of *UISim* for all-pair queries is bounded by $O(|V^o_{\geq 2}||H|^\eta T)$ in expectation, where $T = O\left(\bar{d}^L\left(1 - \sum_{v \in H} d_{in}(v)/m\right)^L\right)$ stands for the expected number of random walks up to length L.

Empirical Analysis. *UISim* is evaluated for all-pair queries (by additive error) and for top-k similarity joins (by Precision@k) but only on small graphs with tens of thousands nodes.

3.2 *FLP & Opt-LP*

Wang et al. [42] propose *SimRank graph*, a modified version of the node-pairs graph (see Figure 1(b)).

DEFINITION 5 (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 slightly 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.*

Based on that, the idea is as follows: *"SimRank is reformalized as the solution of a new linear system, the computation of which can be approximated by Backward Push [28]."* In particular, the linear system is formulated as

$$[I - c(I - Diag(\overrightarrow{vec}(I)))(P^\intercal \otimes P^\intercal)]\overrightarrow{vec}(S) = \overrightarrow{vec}(I), \quad (2)$$

where $\otimes$ denotes the Kronecker product, and $\overrightarrow{vec}(\cdot)$ (resp. $Diag(\cdot)$) transforms a matrix (resp. vector) to the corresponding vector (resp. diagonal matrix). The baseline algorithm, referred to as *ForwardLocalPush (FLP)*, is essentially *Backward Push* [28] for single-target PPR on G^s by setting the virtual node as target and following out-edges instead of in-edges (i.e., *reverse* PPR). An optimized algorithm named *OptimizeLocalPush (Opt-LP)* is proposed to avoid redundant computation and to efficiently handle self-loops. For implementation, the algorithm does not have to materialize G^s. We formalize the above claim as the following lemma[3].

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 $1 - c$:*

$$s(u, v) = \frac{1}{c(1-c)} \pi_{G^s}((u, v), (v_r, v_r)). \quad (3)$$

Theoretical Guarantee. *FLP & Opt-LP* guarantee absolute error following [28], i.e., $|\hat{s}(u, v) - s(u, v)| \leq \varepsilon$ for each (u, v).

Complexity Analysis. We observe that the proposed *average* time complexity $O(\frac{\bar{d}^2}{\varepsilon})$ (Proposition 1 of [42]) is *incorrect* where $\bar{d}$ denotes the average degree of G. This average complexity from [28] is achieved by considering *all possible target nodes* in the graph. However, for all-pair SimRank computation, the target node is *fixed* as the virtual node. We fix the issue using the following lemma.

[2]Note that [52] 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*.

[3]For space constraint, we refer most of the proofs to the full version of the paper [1].

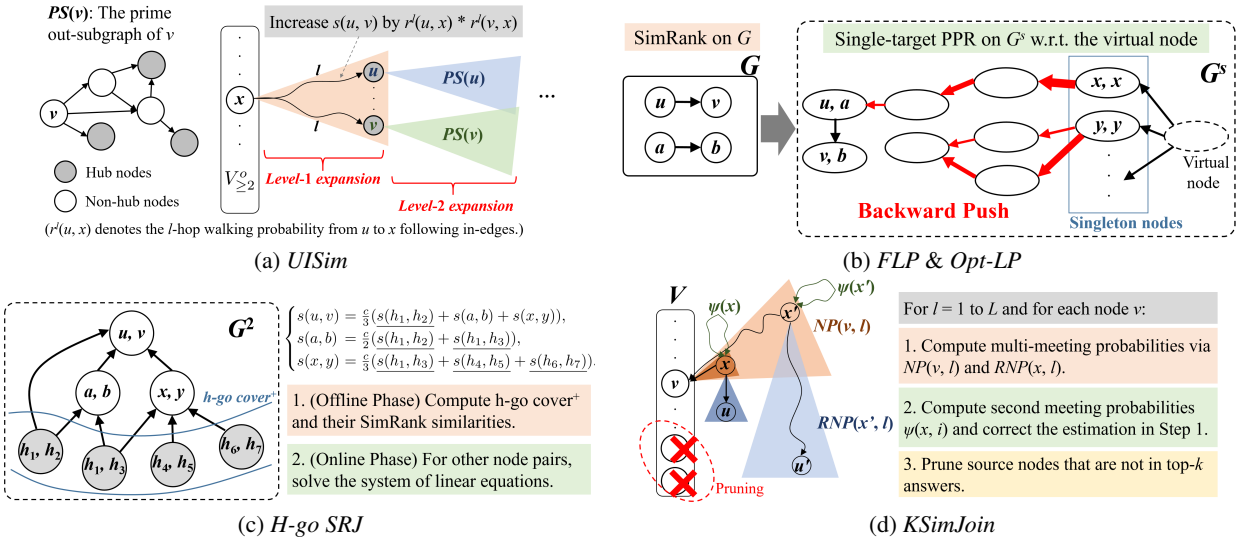


Figure 1: Overview of state-of-the-art algorithms for SimRank-based similarity joins.

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 [52]	✗	N/A	✗	$O(\lvert V^o_{\geq 2}\rvert \lvert H\rvert^\eta T)$	✓	N/A	✓
FLP & Opt-LP [42]	✓	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 [50]	✓	✗	N/A	$\tilde{O}(n\bar{d}^h + \frac{n\lvert H(G)\rvert}{\varepsilon^2})$ (offline), $O(n\bar{d}^{h+3})$ (online)	N/A	Only query time	Only query time
KSimJoin [23]	✓	N/A	✗	$O(n\bar{d}^{\log\frac{1}{\varepsilon}})$	N/A	N/A	Only query time

LEMMA 2. *To achieve an absolute error of ε for all-pair queries, following [28, 38], 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. [42] 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.

3.3 *H-go SRJ*

Inspired by the distance-based upper bound of SimRank which states that two nodes with a large shortest path distance cannot be very similar, the index-based algorithm *H-go SRJ* [50] answers threshold-based similarity joins with the following idea (Figure 1(c)): *"First choose a proper set of node pairs (i.e, the* h-go cover$^+$*) and precompute their SimRank estimations. For any other node pair, its SimRank estimation can be efficiently computed following recursive SimRank definition and with the help of the h-go cover$^+$."*

In the offline phase, the algorithm first computes the h-go cover$^+$ index with similar idea as the vertex cover, so that any h-hop path on the node-pairs graph intersects with the index. 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* conducts graph traversal from it and builds a system of linear equations, which can be solved using Gaussian elimination.

Theoretical Guarantee. [50] does not discuss the error guarantee of *H-go SRJ*. We remedy this issue with the following analysis. Given an error parameter ε, for each indexed node pair we invoke the Monte Carlo algorithm [9] 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 the ε error guarantee also holds for the similarity of any node pair computed in the online phase.*

Complexity Analysis. According to [50], 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 $\tilde{O}(\frac{n\lvert H(G)\rvert}{\varepsilon^2})$ time, where $H(G)$ denotes the h-go cover$^+$ of G. Therefore, the offline phase needs $\tilde{O}(n\bar{d}^h + n\frac{\lvert H(G)\rvert}{\varepsilon^2})$ time. For the online phase, given a node pair, building the linear system by graph traversal needs $O(\bar{d}^{2h})$ time, while solving it incurs $O(\bar{d}^{2h+1})$ time with all optimizations [50]. 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. [50] evaluates query efficiency on graphs with up to millions of nodes but ignores the evaluation of accuracy.

3.4 *KSimJoin*

Based on the random walk interpretation, *KSimJoin* [23] answers top-k similarity joins via decomposing the SimRank value by meeting steps, i.e., $s(u,v) = \sum_{l=0}^{\infty} s^{(l)}(u,v)$, where $s^{(l)}(u,v)$ is the *first meeting probability* of two $\sqrt{c}$-walks from u and v at exact l-th

step. It employs the following key idea: *"Compute the meeting probabilities in ascending order of meeting steps. Multi-meeting cases at each step are eliminated by carefully utilizing previous computation with fewer steps."*

Specifically, the following equation is adopted to compute $s^{(l)}(u,v)$:

$$s^{(l)}(u,v) = \sum_{x \in V} \Pr((u,v) \xrightarrow{l} (x,x)) - \sum_{i \in [1,l-1], y \in V} \Pr((u,v) \xrightarrow{i} (y,y)) \psi(y, l-i), \quad (4)$$

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*. The algorithm performs a breadth-first search (BFS) from every node v following in-edges (Figure 1(d)). 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 finds all node u that is l-step from x, 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 4, it excludes all multi-meeting cases, where the second meeting probabilities $\psi(x,i)$ can be computed following a similar BFS-based incremental approach. For top-k queries, after each iteration of l, *KSimJoin* uses a carefully designed upper bound to prune nodes that cannot be in the top-k answers, where the upper bound is computed by relaxing Equation 4.

Theoretical Guarantee. [23] does not analyze the error guarantee of *KSimJoin*. Our lemma states that *KSimJoin* (without upper bound pruning) is essentially equivalent to the *Power Method* [14].

LEMMA 4. *By considering the terms from $l = 0$ to L with $L = O(\log \frac{1}{\varepsilon})$, 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 [23]. The cost for 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 [23] only focuses on the efficiency of top-k queries on small and medium graphs.

3.5 Comparison of State-of-the-art Algorithms

Apart from the specific interpretation of SimRank in the original papers, *KSimJoin*, *UISim* and *FLP & Opt-LP* can be viewed as different ways of prioritizing meeting probabilities. *KSimJoin* orders random walk pairs by walk length, *FLP & Opt-LP* prioritize them by the walking probabilities, and *UISim* can be considered as an approximation of the strategy in *FLP & Opt-LP*. However, existing experimental study is quite limited, including lack of evaluation on large graphs [23, 42, 52] and for estimation error [23, 50]. More importantly, none of the algorithms provides any approximation guarantee for both types of similarity joins.

4 OUR IMPROVED ALGORITHMS

4.1 Algorithm Framework

We adopt a simple yet effective idea [26, 40] to answer threshold-based and top-k queries with approximation bound (Figure 2). 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. We present specific stopping conditions for our problems.

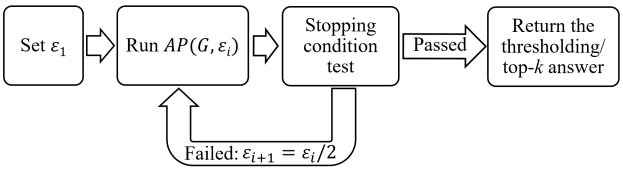


Figure 2: Illustration of the algorithm framework.

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. 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 halve the error parameter for the next iteration (Line 8). 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).

Remark. The existence of approximation bound $\rho \in (0,1)$ is a necessary condition for *APThres* to stop. Without ρ, which means setting $\rho = 1$, *APThres* stops only if $\mathcal{C} = \emptyset$, which does not always hold. In worst cases, we claim that no such an algorithm exists with better efficiency than computing the exact SimRank values. Suppose that a graph has $|\mathcal{R}|$ node pairs with SimRank values being exactly θ, while other (non-singleton) node pairs have zero similarity. Even for a ground truth algorithm that guarantees ε_{min} additive error, it still might return an answer that is arbitrary bad in terms of approximation bound, e.g., when all estimations are below θ.

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

THEOREM 1. *APThres with an all-pair algorithm $\mathcal{AP}$ of absolute error guarantee holds approximation bound ρ for threshold-based similarity joins.*

PROOF (SKETCH). All node pairs included in $\mathcal{R}(\theta)$ belong to the actual result because $\hat{s}_i(u,v) \geq \theta + \varepsilon$ and $|\hat{s}_i(u,v) - s(u,v)| \leq \varepsilon$. Similarly, if some node pair in the real answer is not included, it must be in the candidate set $\mathcal{C}$. Therefore, $\frac{|\mathcal{R}(\theta)|}{|\mathcal{R}(\theta)|+|\mathcal{C}|}$ is a lower bound of the approximation guarantee. The above claims also hold for an algorithm $\mathcal{AP}$ with a small failure probability. If the failure probability of *APThres* is δ (e.g. $o(\frac{1}{n})$), it is sufficient to set the failure probability of $\mathcal{AP}$ as $\delta / \log_2 \frac{1}{\varepsilon_{min}}$, so the theorem holds. □

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** $\mathcal{R}(\theta)$;

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** $\mathcal{R}(k)$;

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 sets 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).

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

THEOREM 2. *APTop-k with an all-pair algorithm $\mathcal{AP}$ of absolute error guarantee holds approximation bound ρ for top-k queries.*

PROOF (SKETCH). We claim when the stopping condition is satisfied, for each $j \leq \lceil \rho k \rceil$ and each $j' \geq k+1$, the SimRank similarity of the j-th node pair is no smaller than that of the j'-th, as the lower bound of the former is no smaller than the upper bound of the latter. Consequently, it means that the top-$\lceil \rho k \rceil$ node pairs are always included in the exact result. Similar to Theorem 1, this claim holds for algorithm $\mathcal{AP}$ with a small failure probability. □

4.2 Randomized Reverse Local Push (R^2LP)

We present an all-pair algorithm following the random walk interpretation and leveraging the local push techniques [28, 38]. It is inspired by the advantage of *Randomized Backward Search (RBS)* [38] over *Backward Push* for PPR. As computing SimRank on the node-pairs graph with $O(n^2)$ nodes incurs much more computational and memory costs than computing PPR on the original graph, to guarantee the practical efficiency, we devise a pruning strategy with absolute error guarantee to dramatically enhance query time.

Algorithm Description. We refer to the algorithm as *Randomized Reverse Local Push* (R^2LP) (see Algorithm 3). 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 meeting 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. Following previous work [27, 50], by only considering terms up to $s^{(L)}(u,v)$, the incurred error is bounded by c^{L+1}. Therefore, it is sufficient to set $L = O(\log \frac{1}{\varepsilon})$ to bound an $O(\varepsilon)$ error (Line 1). We initialize the algorithm in Lines 2-3, where $\hat{s}^{(l)}(u,v)$ corresponds to the reserves of PPR [28, 38], and we omit the residues because it holds that $r^{(l)}(u,v) = \hat{s}^{(l)}(u,v)/(1-c)$. Then, for $l = 0$ to $L-1$, we compute the estimates $\hat{s}^{(l+1)}(u,v)$ based on $\hat{s}^{(l)}(u,v)$ (Lines 4-10). For each $\hat{s}^{(l)}(u,v)$ at level l, we check if its value is large enough (Line 5). For now, we set $f(\varepsilon) = 0$ 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). For node pair (u', v') with limited amount of in-neighbors in G^s, we conduct deterministic push as in *Backward Push* (Line 7). Otherwise, the randomized push is applied (Lines 8&10), which gives an unbiased estimation [38] but ignores many node pairs in practice. The returned value $\hat{s}(u,v)$ summarizes the estimations of every step l (Lines 11-12).

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 [38] needs the out-neighbors of (u,v) to be sorted in *ascending* order of *in-degree*. This can be achieved by first preprocessing the adjacency lists of G before invoking the algorithm.

Pruning Strategy. Recall that the above algorithm (as well as *RBS*) combines deterministic and randomized push for algorithmic efficiency. Intuitively, it encourages most node pairs to be handled by the randomized process. When applied to SimRank computation, due to the excessive number of node pairs in G^s, the randomized part of the algorithm still needs to be improved to effectively prune the node pairs with negligible similarities. We additionally propose a deterministic pruning strategy to directly rule out estimations with small values at each step l. To be precise, we set $f(\varepsilon) = \tilde{O}(\varepsilon)$ (i.e., we omit the logarithmic factor of ε) in Line 5 of Algorithm 3.

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 $u, v \in V$

$L \leftarrow \log_c(1-c)\varepsilon$;

$\hat{s}^{(l)}(u, v) \leftarrow 0$ for $l \in [0, L]$ and $u, v \in V$;

$\hat{s}^{(0)}(v, v) \leftarrow 1$ for all $v \in V$;

for $l = 0$ *to* $L - 1$ **do**

  for *each* (u, v) *with* $\hat{s}^{(l)}(u, v) > f(\varepsilon)$ **do**

    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**

      $\hat{s}^{(l+1)}(u', v') \leftarrow \hat{s}^{(l+1)}(u', v') + \frac{c\hat{s}^{(l)}(u,v)}{d_{in}(u')d_{in}(v')}$;

    $r \leftarrow rand(0, 1)$;

    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**

      $\hat{s}^{(l+1)}(u', v') \leftarrow \hat{s}^{(l+1)}(u', v') + \frac{(1-c)\varepsilon}{\sqrt{d_{in}(u')d_{in}(v')}}$;

$\hat{s}(u, v) \leftarrow \sum_{l=0}^{L} \hat{s}^{(l)}(u, v)$;

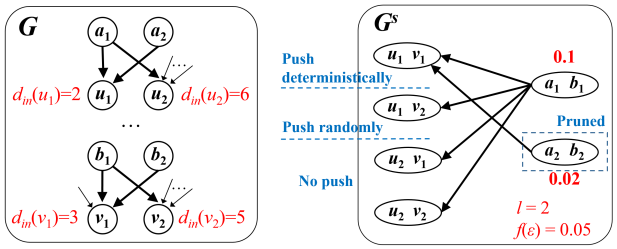
return non-zero $\hat{s}(u, v)$ for $u, v \in V, u \neq v$;

For the deterministic process, it is almost unaffected as the algorithm requires $\hat{s}^{(l)}(u, v) > \frac{(1-c)\varepsilon}{c} \cdot \sqrt{d_{in}(u')d_{in}(v')}$ and the RHS usually exceeds ε. Hence, the deterministic part guarantees the large fraction of the similarity value is computed exactly. For the randomized process, the pruning strategy becomes effective when r is unfortunately very small, namely, $r \leq \frac{c}{(1-c)\sqrt{d_{in}(u')d_{in}(v')}}$. As we shall see in the experiments, the pruning power is significant.

Running Example. Consider the toy example in Figure 3. We have $d_{in}(u_1) = 2, d_{in}(u_2) = 6, d_{in}(v_1) = 3$, and $d_{in}(v_2) = 5$ (Figure 3(a)). Assume that $c = 0.6$, $l = 2$, $\varepsilon = f(\varepsilon) = 0.05$, and we have $\hat{s}^{(2)}(a_1, b_1) = 0.1, \hat{s}^{(2)}(a_2, b_2) = 0.02$. Next we compute $\hat{s}^{(3)}(u_i, v_j)$ as shown in Figure 3(b). Since $\hat{s}^{(2)}(a_1, b_1) > f(\varepsilon)$, we conduct push for all of its out-neighbors in G^s. Note that the threshold for deterministic push is $\frac{c\hat{s}^{(2)}(a_1,b_1)}{(1-c)\varepsilon} = \frac{0.6 \cdot 0.1}{(1-0.6) \cdot 0.05} = 3$. For (u_1, v_1), as $\sqrt{d_{in}(u_1)d_{in}(v_1)} = \sqrt{6} < 3$, we increase $\hat{s}^{(3)}(u_1, v_1)$ by $\frac{c\hat{s}^{(2)}(a_1,b_1)}{d_{in}(u_1)d_{in}(v_1)} = \frac{0.6 \cdot 0.1}{2 \cdot 3} = 0.01$. For other three node pairs $(u_1, v_2), (u_2, v_1)$, and (u_2, v_2), we conduct randomized push since $\sqrt{d_{in}(u_i)d_{in}(v_j)} > 3$. Suppose that $r = 0.75$, then we have $\sqrt{d_{in}(u_1)d_{in}(v_2)} < 3/0.75 = 4$, and $\hat{s}^{(3)}(u_1, v_2)$ is increased by $\frac{(1-c)\varepsilon}{\sqrt{d_{in}(u_1)d_{in}(v_2)}} = \frac{(1-0.6) \cdot 0.05}{\sqrt{2 \cdot 5}} = 0.0063$. Note that we do not conduct any push for (u_2, v_1) and (u_2, v_2). As for (a_2, b_2), we ignore it according to our pruning strategy since $\hat{s}^{(2)}(a_2, b_2) < f(\varepsilon)$.

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.

LEMMA 5. *By conducting the randomized push procedure of R^2LP for $O(\log \frac{n}{\delta})$ times and apply the Median-of-Mean trick [7], we have $|\hat{s}(u, v) - s(u, v)| \leq \varepsilon$ with at least $1 - \delta$ probability.*



(a) G (b) G^s

Figure 3: A running example for R^2LP.

PROOF (SKETCH). Note that without the pruning strategy, by Lemma 1, to guarantee ε absolute error it is sufficient to set the input error parameter to $c(1-c)\varepsilon$. Since without pruning the algorithm is equivalent to invoking *RBS* on G^s, the lemma follows by Theorem 4.2 of [38]. Next we show when conducting *RBS* on G^s, the caused error by pruning is also bounded.

First, we prove that $\mathbb{E}[\hat{\pi}'_{l+1}(u, t)] \geq \pi_{l+1}(u, t) - \frac{1-\alpha}{\alpha}\varepsilon$ for $\forall u \in V$ and $l \in [0, L)$. Here, $\hat{\pi}'$ and $\hat{\pi}$ denote the PPR estimation (on G^s) with and without pruning, respectively. We prove it by induction. For $l = 0$, $\hat{\pi}'_0(t, t) = \alpha$ and $\hat{\pi}'_0(v, t) = 0$ for any $v \neq t$ where t denotes the target node (v_r, v_r) of G^s (we assume $\alpha > \varepsilon$), and the claim holds. Suppose $\mathbb{E}[\hat{\pi}'_l(v, t)] \geq \pi_l(v, t) - \sum_{i=1}^{l-1}(1-\alpha)^i\varepsilon$ holds, then we have $\hat{\pi}'_{l+1}(u, t) = \sum_{v \in N_{out}(u)} X'_{l+1}(u, v)$ and

$$\mathbb{E}[\hat{\pi}'_{l+1}(u, t)|\{\hat{\pi}'_l\}] = \sum_{v \in N_{out}(u)} \mathbb{E}[X'_{l+1}(u, v)|\{\hat{\pi}'_l\}]$$

$$\geq \sum_{v \in N_{out}(u)} \frac{(1-\alpha)\hat{\pi}'_l(v, t)}{d_{out}(u)} - (1-\alpha)\varepsilon. \quad (5)$$

Hence, we can derive that

$$\mathbb{E}[\hat{\pi}'_{l+1}(u, t)] \geq \sum_{v \in N_{out}(u)} \frac{(1-\alpha)\mathbb{E}[\hat{\pi}'_l(v, t)]}{d_{out}(u)} - (1-\alpha)\varepsilon$$

$$\geq \sum_{v \in N_{out}(u)} \frac{(1-\alpha)(\pi_l(v, t) - \sum_{i=1}^{l-1}(1-\alpha)^i\varepsilon)}{d_{out}(u)} - (1-\alpha)\varepsilon$$

$$= \pi_{l+1}(u, t) - \sum_{i=1}^{l}(1-\alpha)^i\varepsilon \geq \pi_{l+1}(u, t) - \frac{1-\alpha}{\alpha}\varepsilon. \quad (6)$$

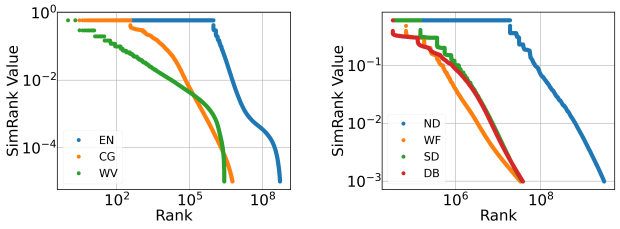
Second, we show that with our pruning strategy, the variance of every $\hat{\pi}'_l(v, t)$ is still bounded by Lemma 4.4 of [38]. The key observation is that the variance comes from the randomized push and pruning reduces its estimation value. Specifically, we have $\mathrm{Var}[\hat{\pi}'_{l+1}(u, t)|\{\hat{\pi}'_l\}] \leq \mathrm{Var}[\hat{\pi}_{l+1}(u, t)|\{\hat{\pi}_l\}]$.

By the above discussion, we have $\mathbb{E}[\hat{\pi}'(u, t)] \geq \pi(u, t) - L\frac{1-\alpha}{\alpha}\varepsilon$, and the variance of $\hat{\pi}'(u, t)$ is also bounded by $L\alpha\varepsilon^2$ [38]. By setting $L = O(\log \frac{1}{\varepsilon})$ to bound the truncation error by $O(\varepsilon)$, the total estimation error can be bounded by ε, and the lemma follows. □

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

THEOREM 3. *The expected time complexity of R^2LP is bounded by $\tilde{O}(\frac{\sum_{u \in V}\sum_{v \in V}\sqrt{d_{in}(u)d_{in}(v)}s(u,v)}{\varepsilon})$.*

PROOF (SKETCH). It follows directly from Lemma 5 and Theorem 4.2 of [38]. □



(a) Small graphs (b) Medium graphs

Figure 4: Distribution of all-pair SimRank values.

5 COMPUTATIONAL COMPLEXITY ANALYSIS

5.1 Modeling SimRank Distribution

We demonstrate in Figure 4 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, 6], we propose the following assumption of SimRank distribution.

DEFINITION 6 (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 $x = 0, \ldots, \lfloor \log_2 nnz \rfloor$, there exist parameters $\beta > 0$ and $b_2 > b_1 > 0$ such that*

$$\sum_{j=2^x}^{2^{x+1}-1} b_1 \cdot r(j)^{-\beta} \leq \sum_{j=2^x}^{2^{x+1}-1} s_j \leq \sum_{j=2^x}^{2^{x+1}-1} b_2 \cdot r(j)^{-\beta}, \quad (7)$$

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 algorithm framework which integrates an all-pair algorithm with absolute error guarantee (e.g., *Opt-LP* or *R^2LP*).

LEMMA 6. *Given a graph G and an all-pair algorithm $\mathcal{AP}$ that guarantees absolute error, assume that APThres 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 $O(cost(\mathcal{AP}(G, \varepsilon_t)))$.*

Next, we bound the error parameter ε_t for threshold-based queries with the help of Definition 6.

LEMMA 7. *We are given the problem input as in Lemma 6. Moreover, the graph G has PLB SimRank distribution. For threshold-based queries with approximation bound ρ, we have*

$$\varepsilon_t = O(\max(\frac{(b_1 - \rho^\beta b_2)\theta}{b_1 + \rho^\beta b_2}, \varepsilon_{min})). \quad (8)$$

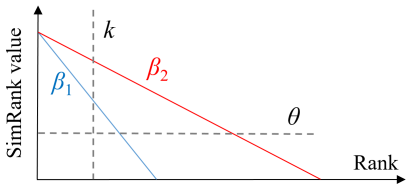


Figure 5: Illustration of the complexity of threshold-based and top-k similarity joins.

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 4. *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 5. *We are given the problem input as in Lemma 7. 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*

$$\varepsilon'_t = O(\max(b_1(x_1 - t)^{-\beta} - b_2(x_2 - t)^{-\beta}, \varepsilon_{min})). \quad (9)$$

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

$$\varepsilon'_t = O(\max((b_1(\lceil \rho k \rceil - t)^{-\beta} - b_2(k+1-t)^{-\beta}), \varepsilon_{min})). \quad (10)$$

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 5. For threshold-based queries, a smaller β (e.g. β_2) leads to a smaller ε_t in Lemma 7, 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 5 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 5, where the k-th SimRank value becomes smaller (also note that the figure is in log-log scale).

6 EXPERIMENTAL EVALUATION

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. All graphs are obtained from [4, 5, 18, 20] with basic statistics listed in Table 3, where $\bar{d}$ is the average degree and β is the fitted PLB exponent of all-pair SimRank distribution[1].

Table 3: Datasets and their statistics.

Dataset	Type	n	m	$\bar{d}$	β
ca-GrQc (CG)	U	5.2K	14.5K	2.77	1.063
Wiki-Vote (WV)	D	7.1K	103.7K	14.57	0.495
email-Enron (EN)	U	36.7K	183.8K	5.01	1.303
Slashdot0922 (SD)	D	82.2K	948.5K	11.54	1.098
DBLP (DB)	U	317.1K	1.05M	3.31	0.866
Wikilinks-fy (WF)	D	65.6K	1.07M	16.35	1.098
Notre Dame (ND)	D	325.7K	1.5M	4.6	0.964
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	-

Ground truths. For small and medium graphs, we compute the all-pair ground truth using *Power Method* [14] with an absolute error about 10^{-7}. For large graphs, since *Power Method* is computationally intractable, we randomly choose 100 nodes in each graph and use *ExactSim* [39] to compute their single-source ground truths with $\varepsilon_{min} = 10^{-7}$. The partial-pair ground truth is used for evaluation.

Baselines. We include *FLP* & *Opt-LP* [42], *UISim* [52], and *KSimJoin* [23] as our baselines. The implementation of [42, 52] is obtained from the authors. We implement other baselines and our improved algorithms in C++. Specifically, we do not consider *Hgo SRJ* [50] for efficiency issues, which shows 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. Throughout the paper, all reported query times are in seconds.

6.2 Evaluation of Additive Error

Experimental settings. For *FLP*, *Opt-LP*, *R²LP* and *KSimJoin* with absolute error guarantee, we vary the error parameter $\varepsilon = \{0.05, 0.01, 0.005, 0.001\}$ for small and medium graphs, and set $\varepsilon = \{0.1, 0.05, 0.01\}$ for large graphs. In particular, we slightly modify *KSimJoin* to answer all-pair queries by excluding the upper bound computation and the iterative pruning framework, and set $L = 2$. For *UISim*, we fix the number of iterations $\eta = 2$ and vary the number of hub nodes $|H| = \{0.2n, 0.5n\}$. In 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 $stopRea$. 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.

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 $\theta = 10^{-3}$ in our experiments.

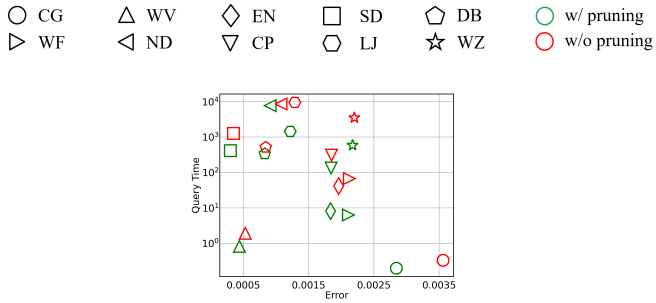


Figure 7: R^2LP with and without the pruning strategy.

The results are demonstrated in Figure 6. For *UISim*, the unfilled and filled markers correspond to $stopRea$'s default values (0.001 and 0.01) and enlarged values (0.005 and 0.05), respectively. Surprisingly, although *UISim* does not guarantee the absolute error, it outperforms *FLP* and *Opt-LP* on 6 of 10 graphs. However, we note that varying the number of hubs has limited influence, that is, the two unfilled markers (as well as the two filled markers) are very close. The reason is that most of graph expansions terminate because of stopping reachability rather than encountering hub nodes.

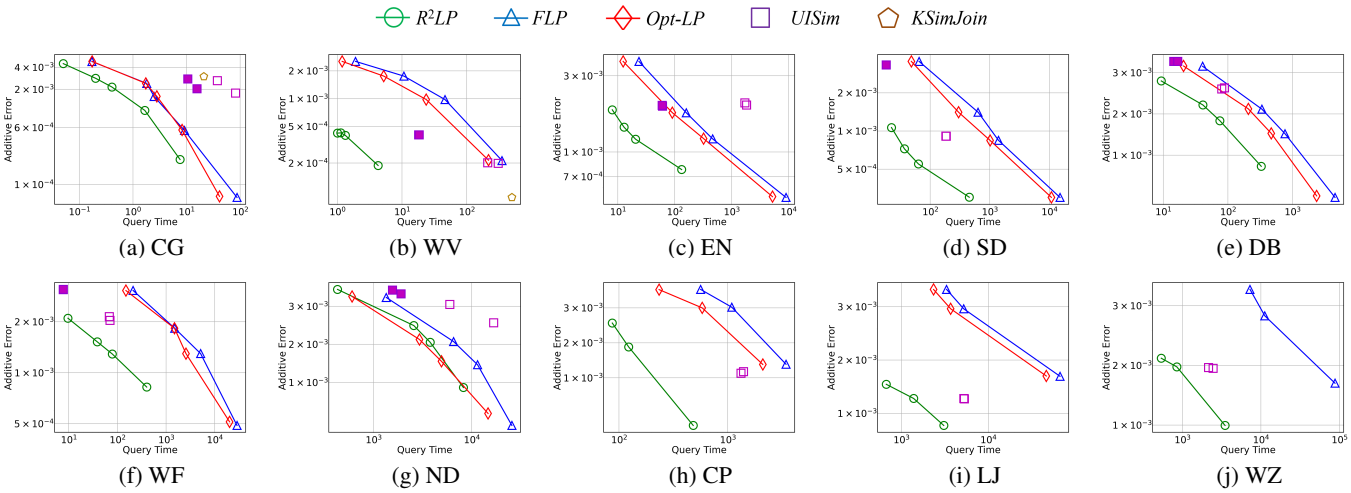
For algorithms with absolute error guarantees, *Opt-LP* outperforms *FLP* on most graphs thanks to the optimizations, except that it fails on the largest graph WZ (not terminated for over 10 hours). *KSimJoin* only manages to answer all-pair queries on the two smallest graphs and has similar performance 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. With the randomized push strategy, *R²LP* 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 *R²LP* and *Opt-LP* show comparable results. Since ND has numerous node pairs of very large similarity values (see Figure 4(b)), the cost of deterministic push dominates.

Evaluation of R^2LP's pruning strategy. Figure 7 demonstrates the performance of *R²LP* with and without the pruning strategy. For each dataset, we tune both settings to ensure comparable additive errors and compare the query time. *R²LP* w/ pruning clearly achieves better effectiveness-efficiency tradeoff. On EN, WF, LJ, and WZ, the speedup is about one order of magnitude.

To better illustrate the pruning power, we use Table 4 to show the number of edge traversals for deterministic and randomized pushes of *R²LP* under both settings. For deterministic push, the costs are very close, while our pruning strategy manages to eliminate up to 47%-96% of the costs for randomized push. The pruning strategy also effectively reduces memory cost, as without pruning the algorithm causes out-of-memory (OOM) error on LJ.

6.3 Evaluation of Threshold-based Queries

Experimental settings and metrics. 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, 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* $= 2/(\frac{1}{Precision} + \frac{1}{Recall})$.



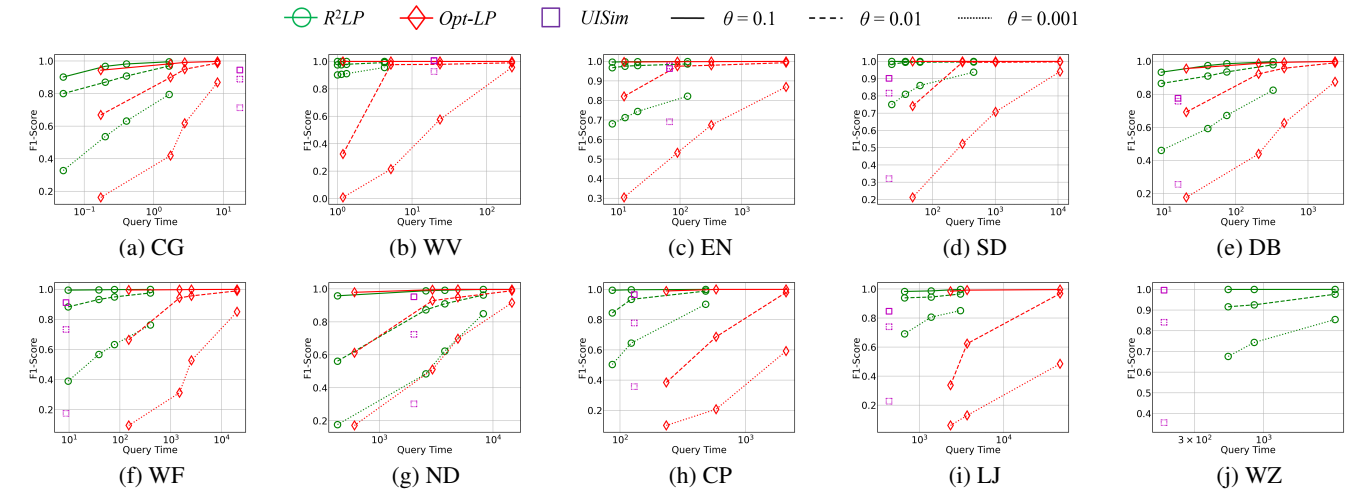
(a) CG (b) WV (c) EN (d) SD (e) DB

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

Figure 6: Additive error of SimRank estimation.

Table 4: R^2LP's pruning power for deterministic and randomized push in terms of edge traversal ($\varepsilon = 0.01$).

		CG	WV	EN	SD	DB	WF	ND	CP	LJ	WZ
Deterministic	w/ pruning	4.29e5	6.82e6	2.56e7	5.69e7	3.49e7	3.98e7	1.49e9	4.81e8	2.65e9	2.03e9
	w/o pruning	4.32e5	6.82e6	2.57e7	5.72e7	3.51e7	3.99e7	1.7e9	4.82e8	OOM	2.03e9
Randomized	w/ pruning	8.83e5	3.22e5	1.18e7	2.29e7	5.85e7	7.98e7	5.74e8	4.68e8	1.43e9	4.5e9
	w/o pruning	3.11e6	5.33e6	2.13e8	6.18e8	3.85e8	4.18e8	4.36e9	8.91e8	OOM	1.85e10



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

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

Figure 8: F1-score for threshold-based queries. No approximation bound is provided.

We first compare the three best all-pair algorithms R^2LP, *Opt-LP*, and *UISim* with $\theta = \{0.1, 0.01, 0.001\}$. 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 8[4]. First, 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 in Section 6.2. On a majority of graphs, it surpasses *Opt-LP* mainly due to the fast query time. However, as we decrease $stopRea$ or increase η, the algorithm fails to compute an output in several 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 5. Consistent with our theoretical guarantee, both *APThres*+*Opt-LP* and *APThres*+R^2LP

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

Table 5: 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

hold the approximation bound on all graphs and all settings of ρ. *APThres+R²LP* achieves similar *F1-score* with much less query time except on ND, of which the reasons are explained in Section 6.2.

However, compared to Figure 8, both algorithms incur significantly more time, and fail on LJ for different reasons[5] while *APThres+Opt-LP* also fails on WZ. Recall that *R²LP* is able to handle LJ with acceptable accuracy if we do not force an approximation bound. Meanwhile, the actual *F1-scores* shown in Table 5 far exceed the bound ρ. We believe this phenomenon is attributed to the pessimistic estimation of the absolute error. In practice, we have $|\hat{s}(u,v) - s(u,v)| \ll \varepsilon$. 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 *Precision@k* $= |R_{\mathcal{A}}(k) \cap R^*(k)|/k$. Because the numbers of node pairs with large similarities vary significantly for different graphs, instead of setting different k for our datasets, 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. 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 $|B| = 10^3$. The query nodes are sampled at random.

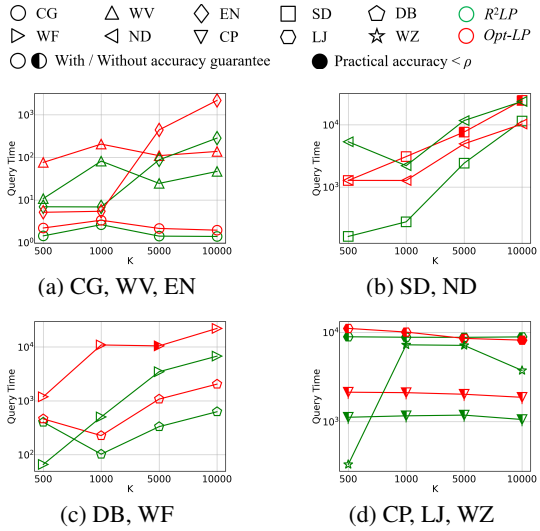
We first compare the state-of-the-art algorithms by fixing $k = 5000$ and $\rho = 0.9$ with the results listed in Table 6. For *UISim*, we fix $\eta = 2$ and set *stopRea* as 10^{-4} for CG and WV, 10^{-3} for EN, and 10^{-2} for all other datasets. Although being fast, *UISim* mainly suffers from accuracy issues. On the other hand, *KSimJoin* has severe scalability problem on medium and large graphs in spite of its pruning strategy. We integrate *Opt-LP* and *R²LP* with *APTop-k* to answer top-k queries. For both of them, the query accuracy is guaranteed. *APTop-k+R²LP* is the fastest on a majority of graphs.

Next, we evaluate the query efficiency w.r.t. k (Figure 9). In most cases, *APTop-k+R²LP* outperforms *APTop-k+Opt-LP* by a significant margin. On three large graphs CP, LJ, and WZ, both

[5]We say an algorithm is time out if it takes over an hour in one iteration.

Table 6: Precision@5000 and query time ($\rho = 0.9$). The practical accuracy is underlined if it falls below ρ.

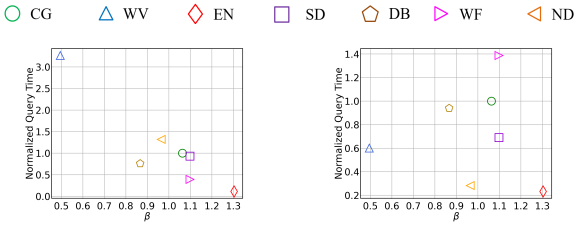
	UISim		*Opt-LP*		*KSimJoin*		*R²LP*	
	Time	P@k	Time	P@k	Time	P@k	Time	P@k
CG	82.5	0.935	2.16	0.999	6.3	0.926	1.42	0.998
WV	308.2	0.992	110.3	0.999	405.9	0.994	24.6	0.998
EN	1189	0.993	443.7	0.997	OOM		88.0	0.996
SD	185.4	0.922	7682	0.996	OOM		2424	0.997
DB	84.0	0.777	1082	0.998	OOM		333.7	0.995
WF	56.0	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 9: Query time vs. k, with $\rho = 0.9$.



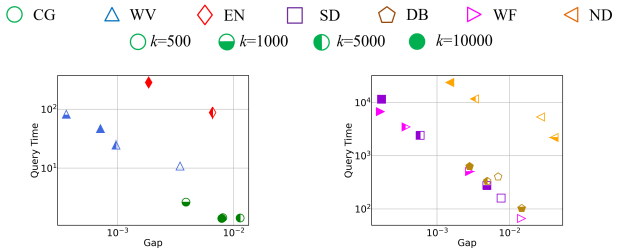
(a) Threshold queries (b) Top-k queries

Figure 10: Query time vs. β.

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 accuracy below ρ on LJ with $k = 10000$ and even fails on WZ.

6.5 Empirical Study of Computational Complexity

We conduct two sets of experiments to validate our theoretical analysis w.r.t. graph properties. As *R²LP* and *Opt-LP* demonstrate similar performance in the above experiments, we only use *R²LP* in the



(a) Small graphs (b) Medium Graphs

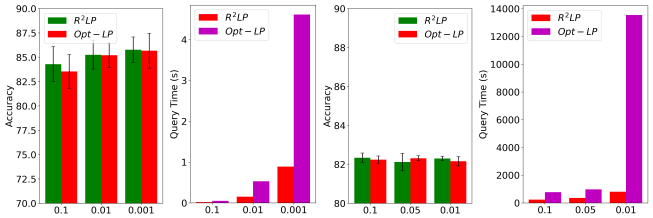
Figure 11: Query time vs. $s_{\lceil \rho k \rceil} - s_{k+1}$ for top-k queries.

following analysis. 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 w.r.t. β, the PLB exponent of SimRank distribution, we first make sure the *practical accuracy* of R^2LP slightly surpasses $\rho = 0.9$ by adjusting the error parameter ε. We then normalize the query time by approximating the numerator of R^2LP's complexity (see Theorem 3) and rescale the normalized values so that the value of CG equals 1. The normalized query time (Figure 10) only reflects the impact of ε, which is mainly affected by both ρ (fixed in our experiment) and graph properties, i.e., β, b_1 and b_2. Generally speaking, the time is in negatively (resp. positively) correlated with β for threshold-based (resp. top-k) queries, which coincides with the derived computational complexity (Theorem 4 & 5). For top-k queries, the data points are more scattered because the impact of β is more complicated as shown by Theorem 5. Therefore, an additional analysis is conducted.

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 11 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 in Figure 9 that query time does not always increase with k.

6.6 Case Study

We adopt a very recent study [24] of graph neural networks (GNNs) to verify the effectiveness of our methods, where SimRank is treated as a new interpretation of GNNs. Instead of iteratively conducting neighborhood aggregation [17] which can be simplified as computing the Personalized PageRank matrix [3, 12], SIMGA [24] precomputes the SimRank matrix and applies it for aggregation. Specifically, we have $Z = softmax(S \cdot f_\theta(F, A))$, where $Z \in \mathbb{R}^{n \times d}$ is the embedding matrix, S is the SimRank matrix, and f_θ is a mapping function (e.g., an MLP) to transform node features F and adjacency matrix A into embeddings. SIMGA uses *Opt-LP* to compute S, and generally outperforms sophisticated GNNs on both homophily and heterophily graphs in terms of classification accuracy. We substitute *Opt-LP* with R^2LP and demonstrate in Figure 12 the accuracy of SIMGA and the cost of computing S on a small dataset Cora ($n = 2.7$K, $m = 5.3$K) and a large dataset Pokec ($n = 1.6$M, $m = 30.6$M). It turns out that the accuracy is comparable to state-of-the-art GNNs [24], e.g., 82.30 ± 0.12 on Pokec. Meanwhile, our method achieves several times of speedup.



(a) Cora (A) (b) Cora (T) (c) Pokec (A) (d) Pokec (T)

Figure 12: A case study of our approach on GNN. "A" stands for accuracy and "T" stands for query time.

7 OTHER RELATED WORK

The *Power Method* [14] is used to compute the ground truth SimRank values on small graphs (e.g., with tens of thousands of nodes), which guarantees a very small absolute error. To accelerate it on larger graphs, [27, 47] improves the algorithmic complexity by pruning redundant computation. Another line of work [13, 21, 44] studies all-pair SimRank computation on dynamic graphs. Several all-pair algorithms [10, 43–46] rely on a modified definition of SimRank, i.e. $S = cP^\intercal SP + (1-c)I$, which results in estimations without absolute error guarantee. A very recent study [48] employs divide-and-conquer to enhance state of the art [42], but is only able to improve the practical efficiency. Their contribution is orthogonal to ours and the implementation can be integrated naturally.

Most of the existing work for SimRank (e.g., [15, 19, 25, 26, 33, 34, 36, 39]) focuses on single-source settings, including *single-source SimRank computation* [15, 33, 34, 36, 39], *single-source top-k queries* [19, 25, 26], and *single-source thresholding queries* [26]. Various algorithms have been proposed based on random walk sampling [15, 33], graph traversal [19, 25, 26, 36], and local push [34]. Besides, a single-pair query for two nodes u and v computes an estimation of $s(u, v)$ with additive error up to ε. Recent algorithms [41] can answer single-pair queries within milliseconds for $\varepsilon \geq 10^{-4}$.

There have been a few survey and benchmark papers for SimRank computation [39, 49], where [49] focuses on single-source top-k queries and [39] computes single-source ground truth on large graphs. Our algorithm framework can be used to evaluate the approximation accuracy and query efficiency of similarity join algorithms.

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] Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2464–2473.

[4] 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*. 587–596.

[5] 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.

[6] 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.

[7] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*. Springer, 693–703.

[8] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. 2009. Power-law distributions in empirical data. *SIAM review* 51, 4 (2009), 661–703.

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

[10] 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.

[11] 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.

[12] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).

[13] 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.

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

[15] 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.

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

[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

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

[19] 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.

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

[21] 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.

[22] 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.

[23] 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

[24] Haoyu Liu, Ningyi Liao, and Siqiang Luo. 2023. SIMGA: A Simple and Effective Heterophilous Graph Neural Network with Efficient Global Aggregation. *arXiv preprint arXiv:2305.09958* (2023).

[25] 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).

[26] 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.

[27] 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

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

[29] 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.

[30] 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.

[31] Takanori Maehara, Mitsuru Kusumoto, and Ken-ichi Kawarabayashi. 2014. Efficient simrank computation via linearization. *arXiv preprint arXiv:1411.7228* (2014).

[32] Takanori Maehara, Mitsuru Kusumoto, and Ken-ichi Kawarabayashi. 2015. Scalable simrank join algorithm. In *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 603–614.

[33] 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.

[34] 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

[35] 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

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] Liangfu Zhang, Cuiping Li, Xue Zhang, and Hong Chen. 2023. Hierarchical All-Pairs SimRank Calculation. In *International Conference on Database Systems for Advanced Applications*. Springer, 252–268.

[49] Zhipeng Zhang, Yingxia Shao, Bin Cui, and Ce Zhang. 2017. An experimental evaluation of SimRank-based similarity search algorithms. *Proceedings of the VLDB Endowment* 10, 5 (2017), 601–612.

[50] 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

[51] 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

[52] 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).

9 APPENDIX

9.1 Proofs

Proof of Lemma 1. Since SimRank has unique solution [14], we only need to prove that the RHS of Equation 3 is a solution of Equation 1. For each singleton node $(v, v) \in G^s$, we have $\pi_{G^s}((v, v), (v_r, v_r)) = c(1-c)$. To be precise, since (v_r, v_r) is the only in-neighbor of (v, v), the probability that a random walk (with stopping probability $1-c$) starts from (v, v) and ends at (v_r, v_r) is $c(1-c)$. It follows that $\frac{1}{c(1-c)}\pi_{G^s}((v, v), (v_r, v_r)) = 1$. Next, we consider non-singleton node (u, v). Recall that the reverse PPR is defined as $\pi(s, t) = (1-a)\sum_{v \in O(t)} \frac{\pi(s,v)}{d_{in}(v)} + a_{s=t}$. Moreover, it holds that $\pi(s, t) = \frac{1-a}{d_{in}(s)} \sum_{v \in I(s)} \pi(v, t) + a_{s=t}$ [28]. Therefore, for $u \neq v$ we have

$$\pi_{G^s}((u, v), (v_r, v_r)) = \frac{c}{|I'(u, v)|} \sum_{(x,y) \in I'(u,v)} \pi_{G^s}((x, y), (v_r, v_r)), \tag{11}$$

where $I'(u, v)$ is the in-neighbors of node (u, v). This is exactly the definition of SimRank for non-singleton nodes. Hence, the RHS of Equation 3 satisfies Equation 1, and the lemma follows. □

Proof of Lemma 2. Recall the time complexity of *Backward Push* [28] for PPR computation w.r.t. the target node t. With an error parameter ε, each push of node v converts at least $\alpha\varepsilon$ residue to v's reserve, where α is the stopping probability. Since each push has $O(d_{in}(v))$ cost, and the number of pushes for v is bounded by $\frac{\pi(v,t)}{\alpha\varepsilon}$. Therefore, the complexity of *Backward Push* is bounded by $O(\frac{\sum_{v \in V} \pi(v,t) d_{in}(v)}{\varepsilon})$. Note that we assume the target node t is given (i.e., fixed), so we do not compute the expected complexity by randomly choosing the target. According to the similar analysis in [38] which considers the increment of residues for v's in-neighbors, the complexity is also bounded by $O(\frac{\sum_{v \in V} \pi(v,t) d_{out}(v)}{\varepsilon})$. Since all-pair SimRank on G is equivalent to reverse PPR on G^s w.r.t. the virtual node (v_r, v_r), the time complexity is bounded by $O(\frac{\sum_{(u,v) \in V_{G^s}} \pi_{G^s}((u,v),(v_r,v_r)) d_{out}(u,v,G^s)}{\varepsilon})$, and is equivalent to $O(\frac{\sum_{u,v \in V} d_{out}(u) d_{out}(v) s(u,v)}{\varepsilon})$. Following [38], it is also bounded by $O(\frac{\sum_{u,v \in V} d_{in}(u) d_{in}(v) s(u,v)}{\varepsilon})$. □

Proof of Lemma 3. Suppose that we solve a system of linear equations with k unknown values $x_1, \ldots, x_k$, and the h-go cover$^+$ is represented as $s_1, \ldots, s_{k'}$. We have $x_i = \frac{c}{d(i)}(\sum_j A_{ij} x_j + \sum_{j'} A_{ij'} s_{j'})$ where the summation has $d(i)$ terms. In the offline phase, we have an estimated value of s_i, denoted as $\hat{s}_i$, with $|\hat{s}_i - s_i| \leq \varepsilon$ for $i \in [1, k']$. In the online phase, let $i = \text{argmax}_j |x_j - \hat{x}_j|$, where the $\hat{x}_j$s are computed from the $\hat{s}_j$s. We have $|x_i - \hat{x}_i| \leq \frac{c}{d(i)}(\sum_j A_{ij}|x_j - \hat{x}_j| + \sum_{j'} A_{ij'}|s_{j'} - \hat{s}_{j'}|) \leq \frac{c}{d(i)} \cdot d(i)\varepsilon \leq \varepsilon$, and the lemma follows. □

Proof of Lemma 4. By eliminating $s^{(l)}(u, v)$ for $l \in [L+1, \infty)$, the estimation error of *KSimJoin* is bounded by c^{L+1} [27]. Thus it is sufficient to set $L = O(\log \frac{1}{\varepsilon})$ to achieve $O(\varepsilon)$ error. □

Proof of Theorem 1. First, note that all node pairs included in $\mathcal{R}(\theta)$ belong to the actual result. Since we have $\hat{s}_i(u, v) \geq \theta + \varepsilon$ and $|\hat{s}_i(u, v) - s(u, v)| \leq \varepsilon$, it follows that $s(u, v) \geq \theta$. Second, if some node pair is missed, it must be in the candidate set $\mathcal{C}$. To be precise, if $s(u', v') \geq \theta$, then we have $\hat{s}(u', v') \geq \theta - \varepsilon$ given the absolute error guarantee. Therefore, $\frac{|\mathcal{R}(\theta)|}{|\mathcal{R}(\theta)|+|\mathcal{C}|}$ is a lower bound of the actual approximation guarantee for that $\mathcal{C}$ might contains node pairs with similarities below θ.

The above claims also hold for an algorithm $\mathcal{AP}$ with a small failure probability. Assume that we set the failure probability of *APThres* as δ (e.g. $o(\frac{1}{n})$). It is sufficient to set the failure probability of $\mathcal{AP}$ as $\delta / \log_2 \frac{1}{\varepsilon_{min}}$. Then the theorem holds. □

Proof of Theorem 2. We claim when the stopping condition is satisfied, for each $j \leq \lceil \rho k \rceil$ and each $j' \geq k+1$, the SimRank similarity of the j-th node pair is no smaller than that of the j'-th one. This is obvious as the lower bound of the former is no smaller than the upper bound of the latter. More specifically, let (u'_j, v'_j) denote the node pair with the j-th largest estimation value in this round. Here, the prime symbol means that the node pair is not the one with the j-th largest *exact* SimRank value. We do not care the specific round number as the discussion always holds for every round. Then we have

$$s(u'_j, v'_j) \geq \hat{s}_{i,\lceil \rho k \rceil} - \varepsilon_i \geq \hat{s}_{i,k+1} + \varepsilon_i \geq s(u'_{j'}, v'_{j'}), \tag{12}$$

which means that the first $\lceil \rho k \rceil$ node pairs with the largest estimation values are included in the exact result. When the algorithm terminates, we also include the next $k - \lceil \rho k \rceil$ node pairs (with the largest estimation values) because the error parameter is typically a loose upper bound of the actual error, and thus they are likely to be part of the real answer (without theoretical guarantee). Similar to Theorem 1, the above discussion holds for algorithm $\mathcal{AP}$ with a small failure probability. □

Proof of Lemma 5. Since *R^{2}LP* is considered as an extension of *Randomized Backward Search (RBS)* [38] to all-pair SimRank computation, we follow the proofs in [38] for the ease of understanding. That is to say, we view *R^{2}LP* as conducting single-target PPR on the SimRank graph in the following proof. Specifically, we first prove the properties of the expectation and variance with our pruning strategy for single-target PPR estimation, which are the extensions of Lemma 4.3 & 4.4 in [38]. Then our lemma follows for the SimRank problem.

We use $\hat{\pi}'_l(v, t)$ and $\hat{\pi}_l(v, t)$ to denote the PPR estimation of v at l-th step with and without pruning, respectively. Similarly, we use $X'_{l+1}(u, v)$ (resp. $X_{l+1}(u, v)$) to denote the contribution of v to u in the next step with (resp. without) the pruning strategy. We have the following lemma.

LEMMA 8 (BOUNDED BIAS FOR EXPECTATION). *By applying our pruning strategy to* RBS*, it holds that* $\mathbb{E}[\hat{\pi}'_{l+1}(u, t)] \geq \pi_{l+1}(u, t) - \frac{1-\alpha}{\alpha}\varepsilon$ *for* $\forall u \in V$ *and* $l \in [0, L)$.

PROOF. First note that by setting $f(\varepsilon) = 0$, which is the same as *RBS*, we have $\mathbb{E}[\hat{\pi}_{l+1}(u, t)] = \pi_{l+1}(u, t)$. In particular, let $X_{l+1}(u, v)$ denote the increment of $\hat{\pi}_{l+1}(u, t)$ by conducting $Push(v)$. According to Lemma 4.3 of [38], we have $\mathbb{E}[X_{l+1}(u, v) | \{\hat{\pi}_l\}] = \frac{(1-\alpha)\hat{\pi}_l(v,t)}{d_{out}(u)}$. With the pruning strategy, we have $\mathbb{E}[X'_{l+1}(u, v) | \{\hat{\pi}'_l\}] \geq \frac{(1-\alpha)(\hat{\pi}'_l(v,t)-\varepsilon)}{d_{out}(u)}$. Note that if $\hat{\pi}'_l(v, t) \geq f(\varepsilon) \approx \varepsilon$, the increment is the same as *RBS*. Otherwise, the increment is 0 and we have $\hat{\pi}'_l(v, t) - \varepsilon \leq 0$. In both cases, the RHS is a lower bound.

Next, we prove the lemma by the following induction procedure. For $l = 0$, $\hat{\pi}'_0(t, t) = \alpha$ and $\hat{\pi}'_0(v, t) = 0$ for all other vs. For $l = 1$, we have $\mathbb{E}[\hat{\pi}'_1(v, t)] = \pi_1(v, t)$ (we assume $\varepsilon < \alpha$ so there is no pruning). It naturally holds that $\mathbb{E}[\hat{\pi}'_1(v, t)] \geq \pi_1(v, t) - (1-\alpha)\varepsilon$.

Suppose $\mathbb{E}[\hat{\pi}'_l(v,t)] \geq \pi_l(v,t) - \sum_{i=1}^{l-1}(1-\alpha)^i\varepsilon$ holds. Since we have $\hat{\pi}'_{l+1}(u,t) = \sum_{v\in N_{out}(u)} X'_{l+1}(u,v)$, it follows that

$$\begin{aligned}\mathbb{E}[\hat{\pi}'_{l+1}(u,t)|\{\hat{\pi}_l\}] &= \sum_{v\in N_{out}(u)} \mathbb{E}[X'_{l+1}(u,v)|\{\hat{\pi}_l\}] \\ &\geq \sum_{v\in N_{out}(u)} \frac{(1-\alpha)\hat{\pi}'_l(v,t)}{d_{out}(u)} - (1-\alpha)\varepsilon.\end{aligned} \tag{13}$$

Hence, we can derive that

$$\begin{aligned}&\mathbb{E}[\hat{\pi}'_{l+1}(u,t)] \geq \frac{(1-\alpha)\mathbb{E}[\hat{\pi}'_l(v,t)]}{d_{out}(u)} - (1-\alpha)\varepsilon \\ &\geq \sum_{v\in N_{out}(u)} \frac{(1-\alpha)(\pi_l(v,t) - \sum_{i=1}^{l-1}(1-\alpha)^i\varepsilon)}{d_{out}(u)} - (1-\alpha)\varepsilon \\ &= \pi_{l+1}(u,t) - \sum_{i=1}^{l}(1-\alpha)^i\varepsilon \\ &\geq \pi_{l+1}(u,t) - \frac{1-\alpha}{\alpha}\varepsilon.\end{aligned} \tag{14}$$

□

LEMMA 9 (BOUNDED VARIANCE WITH PRUNING). *With our pruning strategy, the variance of every $\hat{\pi}'_l(v,t)$ is still bounded by Lemma 4.4 of [38]. Specifically, for additive error guarantee, it holds that* $\mathrm{Var}[\hat{\pi}'_l(v,t)] \leq \alpha\varepsilon^2$.

PROOF. The key observation is that (1)the variance comes from the randomized push [38] and (2)via pruning, we have

$$\mathrm{Var}[\hat{\pi}'_{l+1}(u,t)|\{\hat{\pi}'_l\}] \leq \mathrm{Var}[\hat{\pi}_{l+1}(u,t)|\{\hat{\pi}_l\}]. \tag{15}$$

Then the lemma holds following the proof of Lemma 4.4 in [38]. □

Per the above lemmas, we have $\mathbb{E}[\hat{\pi}'(u,t)] \geq \pi(u,t) - L\frac{1-\alpha}{\alpha}\varepsilon$, and the variance of $\hat{\pi}'(u,t)$ is also bounded by $L\alpha\varepsilon^2$. By setting $L = O(\log\frac{1}{\varepsilon})$, the truncation error is bounded by $O(\varepsilon)$. To be precise, we can set $L = \log_{1-\alpha}\frac{\varepsilon}{3}$ to achieve a truncation error ε_t of $\frac{\varepsilon}{3}$. The pruning error parameter ε_p is set to $\frac{\varepsilon}{3}\cdot\frac{\alpha}{1-\alpha}$ so that $L\frac{1-\alpha}{\alpha}\varepsilon_p \leq \frac{2\varepsilon}{3}$. The total estimation error is thus bounded by ε. By adopting the Median-of-Mean trick, we can bound the failure probability as in [38]. The above conclusion also holds for all-pair SimRank estimation because of the equivalence to single-target PPR (Lemma 1). □

Proof of Theorem 3. It follows directly from Lemma 5 and Theorem 4.2 of [38]. □

Proof of Lemma 6. According to the *APThres* (or *APTop-k*) framework and with $p > 1$, the computational complexity is bounded by $\sum_{i=1}^{t} cost(\mathcal{AP}(G,\varepsilon_i)) \leq cost(\mathcal{AP}(G,\varepsilon_t))\cdot(1+\frac{1}{p}+\cdots+\frac{1}{p^{t-1}}) = O(cost(\mathcal{AP}(G,\varepsilon_t)))$. □

Proof of Lemma 7. Recall that *APThres* stops when $\frac{|\mathcal{R}(\theta)|}{|\mathcal{R}(\theta)|+|\mathcal{C}|} \geq \rho$. Assume that at t-th iteration the condition is satisfied, we develop the lower bound of $|\mathcal{R}(\theta)|$ and the upper bound of $|\mathcal{R}(\theta)\cup\mathcal{C}|$ (note that $\mathcal{R}(\theta)$ and $\mathcal{C}$ are disjoint). With high probability, the following claims hold. We have $\{(u,v)|s(u,v) \geq \theta + 2\varepsilon_t\} \subseteq \mathcal{R}(\theta)$, as any such node pair (u,v) has $\hat{s}_t(u,v) \geq \theta + \varepsilon_t$. We find the maximum j and x (denoted as j^* and x^* respectively) such that $b_1 r(j^*)^{-\beta} \geq \theta + 2\varepsilon_t$ and $j^* = 2^{x^*+1} - 1$. It follows that $j^* \leq (\frac{b_1}{\theta+2\varepsilon_t})^{\frac{1}{\beta}} + t$. Similarly, we have $\mathcal{R}(\theta)\cup\mathcal{C} \subseteq \{(u,v)|s(u,v) \geq \theta - 2\varepsilon_t\}$. Then we find the minimum j and x (denoted as j' and x' respectively) with $b_2 r(j')^{-\beta} \leq \theta - 2\varepsilon_t$ and $j' = 2^{x'+1} - 1$. It follows that $j' \geq (\frac{b_2}{\theta-2\varepsilon_t})^{\frac{1}{\beta}} + t$. If $j^* \geq \rho j'$, the algorithm must have been stopped at t-th iteration. It is sufficient to have $(\frac{b_1}{\theta+2\varepsilon_t})^{\frac{1}{\beta}} \geq \rho(\frac{b_2}{\theta-2\varepsilon_t})^{\frac{1}{\beta}}$, which is equivalent to $\varepsilon_t \leq \frac{(b_1-\rho^\beta b_2)\theta}{2(b_1+\rho^\beta b_2)}$, on condition that $b_1 - \rho^\beta b_2 > 0$. By setting $\varepsilon_t = O(\max(\frac{(b_1-\rho^\beta b_2)\theta}{b_1+\rho^\beta b_2}), \varepsilon_{min})$, the lemma follows. □

Proof of Theorem 4. It follows directly from Lemma 6 & 7. □

Proof of Theorem 5. To achieve an approximation bound of ρ, it is sufficient to set $s_{\lceil\rho k\rceil} - s_{k+1} \geq 2\varepsilon'_t$ for *APTop-k*. By Equation 7, 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 $s_{\lceil\rho k\rceil} \geq b_1(x_1 - t)^{-\beta}$ and $s_{k+1} \leq b_2(x_2 - t)^{-\beta}$. By setting $\varepsilon'_t = \max(\frac{1}{2}(b_1(x_1-t)^{-\beta} - b_2(x_2-t)^{-\beta}), \varepsilon_{min})$, *APTop-$k$* guarantees the approximation bound with high probability. (Note that the first term may be less than zero, so we need the second term ε_{min}.) By assuming the power law distribution of SimRank and using similar derivation, we have $\varepsilon'_t = \max(\frac{1}{2}(b_1(\lceil\rho k\rceil - t)^{-\beta} - b_2(k+1-t)^{-\beta}), \varepsilon_{min})$. □

9.2 Additional Experiments

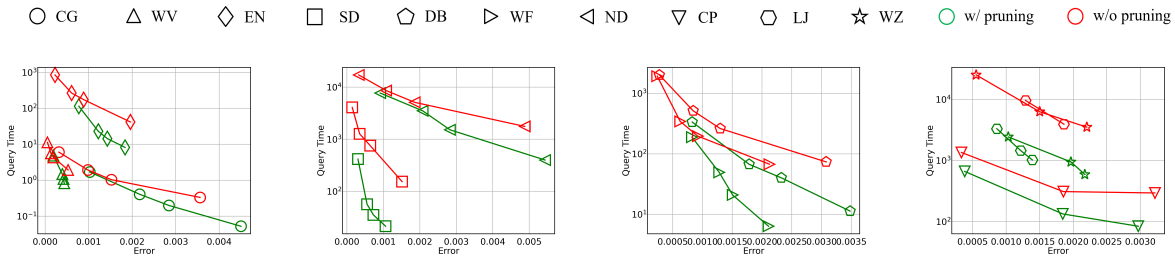
Table 7: Fitted β, b_1, and b_2.

	β	b_1	b_2
CG	1.063	1.543	5.219
WV	0.495	0.709	0.94
EN	1.303	1.909	3.933
SD	1.098	1.459	2.158
DB	0.866	1.347	3.145
WF	1.098	1.501	1.729
ND	0.964	1.697	2.632

Additional experiments for additive error. Note that for the computation of $AvgErr$, We set $\theta = 10^{-3}$ in our experiments for the following reasons. First, applications usually focus on non-negligible SimRank values, which only make up a small portion of the SimRank matrix. Second, 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. Third, we cannot afford to compute and store the ground truths for node pairs with very small similarities for medium and large graphs.

We demonstrate the effectiveness of our pruning strategy. Figure 13 shows the effectiveness-efficiency tradeoff of *R^2LP*. Obviously *R^2LP* w/ pruning achieves better performance.

Additional experiments for threshold-based similarity joins. We show the *Precision* (Figure 14) and *Recall* (Figure 15) separately for the experiments in Figure 8. For *Precision*, first note that *Opt-LP* has *Precision* equal to 1 on all datasets because it always makes underestimations for SimRank values. Surprisingly, *UISim* also underestimates all node similarities on almost all queries (except on ND). The algorithms only considers random walks with limited length in practice, therefore the truncation error outweighs the overestimation by allowing multi-meeting probabilities. As for *R^2LP*, the *Precision* is close to 1 on most small and medium graphs. In



(a) CG, WV, EN (b) SD, ND (c) DB, WF (d) CP, LJ, WZ

Figure 13: Effectiveness-efficiency tradeoff for *R*^{2}*LP* with and without the pruning strategy.

Algorithm 4: Fitting the parameters of PLB distribution

Input: The all-pair ground truth SimRank values $\{s_1, \ldots, s_{nnz}\}$ sorted in descending order
Output: β, b_1, and b_2

1 $b_1^* \leftarrow 0, b_2^* \leftarrow \infty$;
2 **for** $x = x_{min}$ *to* x_{max} **do**
3 Compute $S_x = \sum_{j=2^x}^{\min(2^{x+1}-1, nnz)} s_j$;
4 **for** $\beta = 0.001$ *to* 5 **do**
5 Find maximum b_1 and minimum b_2 with $b_1, b_2 \in [0.001, 50]$ and $b_2 > b_1$, such that $b_1 \sum_j j^{-\beta}$ (resp. $b_2 \sum_j j^{-\beta}$) is the lower (resp. upper) bound of S_x;
6 **if** $b_2/b_1 < b_2^*/b_1^*$ **then**
7 $\beta^* \leftarrow \beta, b_1^* \leftarrow b_1, b_2^* \leftarrow b_2$;
8 Increase β by 0.001;
9 **return** β^*, b_1^*, and b_2^*;

our experiments, following the setting of the baselines, we return all node pairs with SimRank estimations above θ rather than $\theta + \varepsilon$, where the latter also achieves *Precision* close to 1 in practice. As randomness of the algorithm introduces both underestimation as well as overestimation, it also has false positives. On the other hand, *R*^{2}*LP* significantly outperforms *Opt-LP* and *UISim* in terms of *Recall*, since the baselines sacrifice *Recall* for *Precision*. Overall, *R*^{2}*LP* still has a clear advantage over the baselines as demonstrated in Figure 8.

Additional experiments for top-k similarity joins. Table 8 demonstrates the query time and Precision@k of the experiments in Figure 9. Generally speaking, *APTop-k+R*^{2}*LP* is more efficient than *APTop-k+Opt-LP* to achieve comparable accuracy. On 7 of 10 datasets, the speedup is by at least $2\times$.

Additional evaluation of the 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. In particular, Figure 8 and Table 5 imply that a larger approximation bound ρ or a smaller parameter θ leads to more time cost, while Figure 9 directly suggests the time complexity of top-k queries generally increases as we enlarge k.

For our empirical study of the top-k similarity join, it is worthy noting that the observed correlation between the computational complexity and the gap of the $\lceil \rho k \rceil$-th and the $(k+1)$-th largest SimRank values also fits the sampling complexity of the top-k identification problem [11, 16] in the multi-armed bandit setting, where the backward push procedure (such as *R*^{2}*LP* and *Opt-LP*) is considered as generating an estimation for each arm (i.e., node pair) with ε confidence interval.

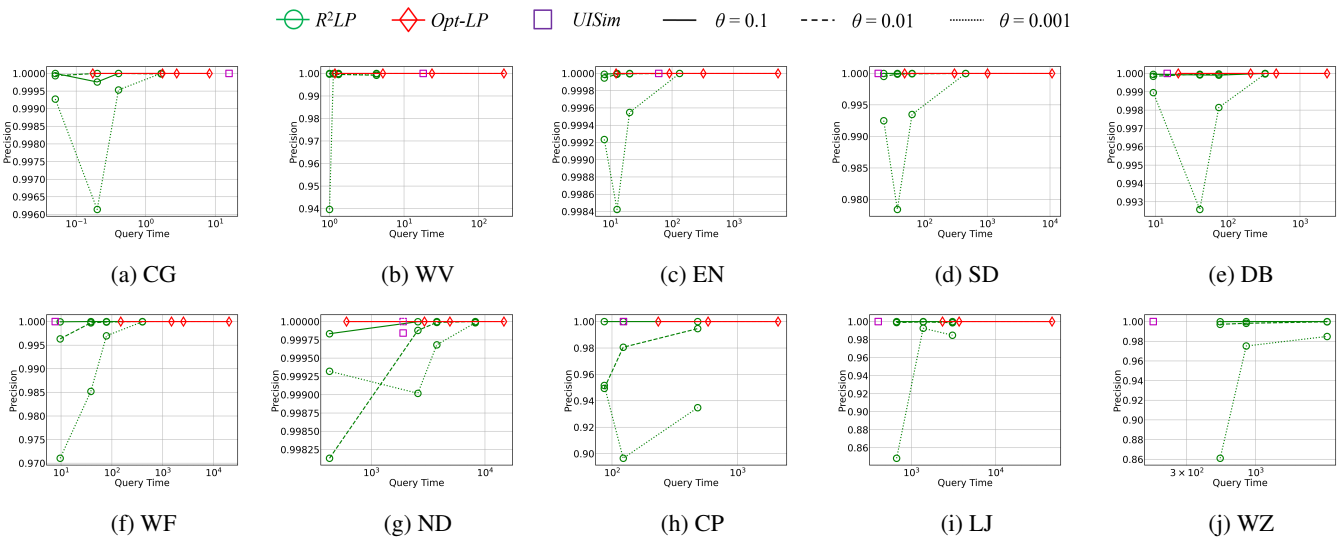
Empirical validation for the PLB distribution of SimRank. We empirically validate that all the small and medium graphs used in our experiments have ground truth SimRank distribution follow PLB. The three large datasets are ignored since we only have the single-source ground truth of 100 query nodes. Note that the definition of PLB is different from the power law distribution, namely, we do not assume the concrete distribution form for the SimRank values. We instead suppose that the distribution is both upper and lower bounded by two power law distributions of the same exponent but with different scaling factors. Therefore, we do not employ the maximum likelihood estimation (MLE) [8] and directly minimize our fitting objective by considering all logarithmic bins.

The pseudocode is illustrated in Algorithm 4. In particular, we empirically observe that in almost all datasets, the all-pair SimRank distribution approximately follow power law for node pairs except those with similarity values significantly above 0.1. (Actually, those node pairs with similarity above 0.1 also follow power law approximately but with a different slope. Besides, they are easy to estimate for most algorithms since two random walks from such a node pair tend to meet after only one or two steps.) Therefore, we focus on the modeling of these node pairs by only considering the bins with $x \in [x_{min}, x_{max}]$, where we set x_{min} as the smallest bin with SimRank value below 0.1, and set $x_{max} = \lfloor \log_2 nnz \rfloor$. In particular, we first compute the average SimRank of each bin (Lines 1-2). Next, we iterate over the possible values of $\beta \in (0, 5]$ with step equal to 0.001 (Lines 4-8). For each value of β, we find the maximum b_1 and minimum b_2 with $b_1, b_2 \in [0.001, 50]$ and $b_2 > b_1$, such that $b_1 \sum_j j^{-\beta}$ (resp. $b_2 \sum_j j^{-\beta}$) is the lower (resp. upper) bound of S_x. Our object is to minimize the ratio between b_2 and b_1.

The fitted values are demonstrated in Table 7. Beside, we also plot the upper and lower PLB bounds of all-pair SimRank distribution arranged in logarithmic bins for small and medium graphs, in Figure 16 & 17 respectively. The results demonstrate the tightness of our fitting method.

Table 8: Query time and Precision@k for $k = \{500, 1000, 5000, 10000\}$ with $\rho = 0.9$. This corresponds to the experiments shown in Figure 9.

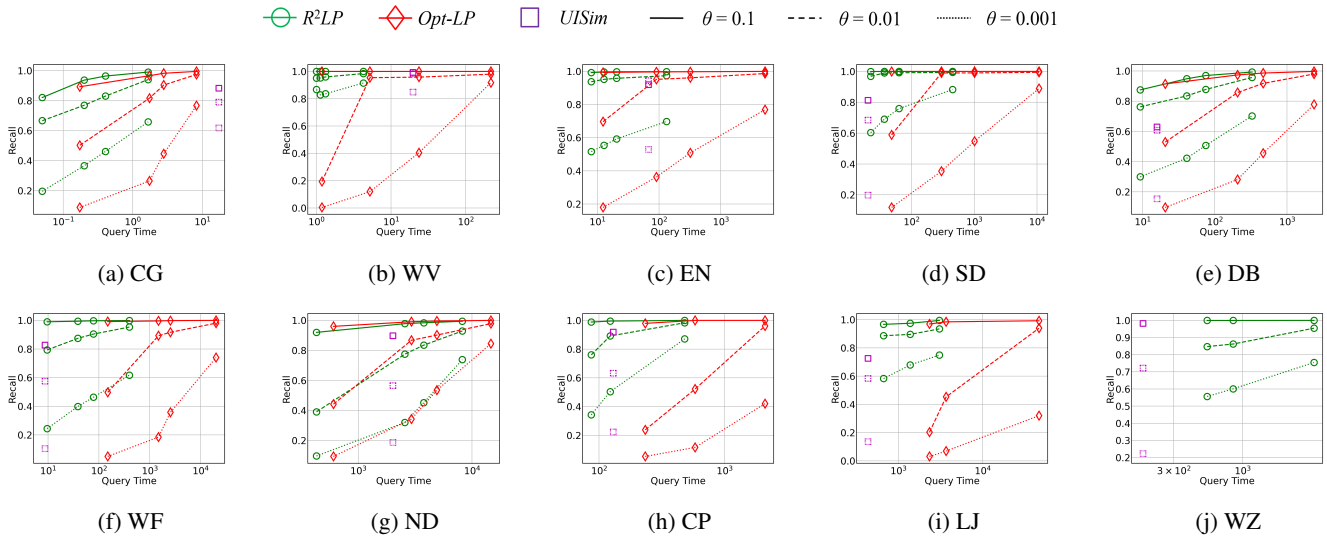
	Opt-LP								*R²LP*							
	k = 500		k = 1000		k = 5000		k = 10000		k = 500		k = 1000		k = 5000		k = 10000	
	Time	P@k	Time	P@k	Time	P@k	Time	P@k	Time	P@k	Time	P@k	Time	P@k	Time	P@k
CG	2.21	0.994	3.36	0.998	2.16	0.999	1.96	0.997	1.45	0.998	2.65	0.997	1.42	0.998	1.40	0.995
WV	76.6	1	206.0	0.999	110.3	0.999	138.6	0.998	10.8	1	82.1	0.999	24.6	0.998	47.0	0.997
EN	5.2	1	5.5	1	443.7	0.997	2185.5	0.998	7.0	1	6.9	1	88.0	0.996	284.9	0.996
SD	1277	0.984	3095	0.997	7682	0.996	24798	0.990	161	0.984	276	0.998	2424	0.997	11564	0.997
DB	465	0.998	228	0.996	1082	0.998	2052	0.998	404	0.996	103	0.994	334	0.995	631	0.996
WF	1197	0.988	10927	0.998	10501	0.99	22157	0.989	66	0.99	506	0.997	3488	0.995	6777	0.994
ND	1281	0.99	1278	0.998	4964	0.997	10336	0.999	5360	0.996	2211	0.996	11749	0.993	23895	0.996
CP	2147	0.986	2122	1	2037	0.992	1882	0.994	1126	1	1168	0.999	1190	0.994	1063	0.979
LJ	11107	0.948	10109	0.973	8586	0.964	8210	0.639	8953	0.988	8845	0.986	8824	0.970	8926	0.947
WZ	OOM								334	1	7277	1	7171	1	3762	0.998



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

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

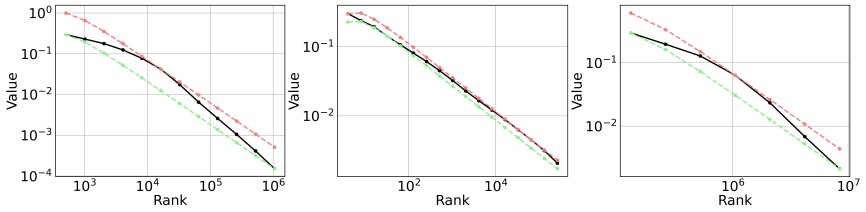
Figure 14: Precision of the experiments in Figure 8 for threshold-based queries. No approximation bound is provided.



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

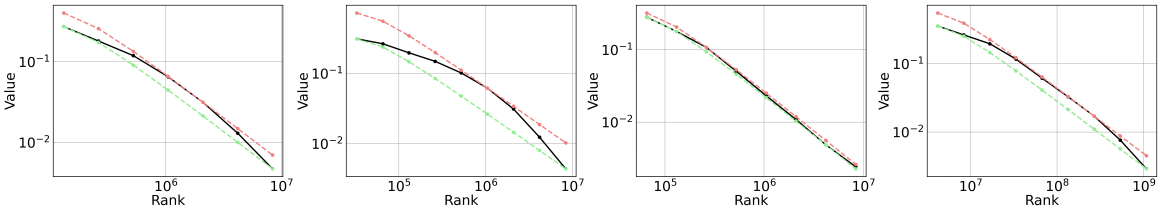
(f) WF (g) ND (h) CP (i) LJ (j) WZ

Figure 15: Recall of the experiments in Figure 8 for threshold-based queries. No approximation bound is provided.



(a) CG (b) WV (c) EN

Figure 16: Upper and lower bound of all-pair SimRank distribution arranged in logarithmic bins (for small graphs).



(a) SD (b) DB (c) WF (d) ND

Figure 17: Upper and lower bound of all-pair SimRank distribution arranged in logarithmic bins (for medium graphs).