Near-Optimality for Single-Source Personalized PageRank*

XINPENG JIANG, Nanyang Technological University, Singapore
HAOYU LIU, Nanyang Technological University, Singapore
SIQIANG LUO, Nanyang Technological University, Singapore
XIAOKUI XIAO, National University of Singapore, Singapore

The Single Source Personalized PageRank (SSPPR) query is a fundamental problem in graph OLAP, and has been extensively studied across various data-driven applications. For a source node s in a graph G, SSPPR measures the PPR score $\pi(s,t)$ for every node $t \in G$, where $\pi(s,t)$ is the probability that an α-decay random walk starting from s terminates at t. Despite decades of research, a significant gap remains between the known. Under two standard accuracy guarantees—*absolute error* (SSPPR-A), requiring $|\hat{\pi}(s,t) - \pi(s,t)| \leq \epsilon$ for all t, and *relative error* (SSPPR-R), requiring $|\hat{\pi}(s,t) - \pi(s,t)| \leq c \cdot \pi(s,t)$ for all t with $\pi(s,t) \geq \delta$—the best-known upper bounds by [6, 39, 43] collectively yield $O\Big(\min(\log(1/\epsilon)/\epsilon^2, \sqrt{m \log n}/\epsilon, m\log(1/\epsilon))\Big)$ for SSPPR-A and $O\Big(\min\Big(\log(1/n)/\delta, \sqrt{m\log(n/\delta)}, m\log(\frac{\log(n)}{m\delta})\Big)\Big)$ for SSPPR-R (n, m denote the number of nodes and edges). Meanwhile, only trivial lower bounds are known—$\Omega(\min(n, 1/\epsilon))$ [42] and $\Omega(\min(n, 1/\delta))$—leaving a substantial theoretical gap yet to be closed. This work narrows or even closes this gap. On the upper bound side, we tighten the computational complexities of the SSPPR-A and SSPPR-R queries to $O(1/\epsilon^2)$ and $O\Big(\min\Big(\log(1/\delta)/\delta, m + n\log(n)\log(\frac{\log(n)}{m\delta})\Big)\Big)$, respectively. These results improve upon the best-known bounds by factors of $\log(1/\epsilon)$ and $\log(\frac{\log(n)}{m\delta})$. On the lower bound side, under the standard arc-centric model, we establish significantly stronger results: $\Omega(\min(m, 1/\epsilon^2))$ for SSPPR-A (improving by m/n or $1/\epsilon$) and $\Omega(\min(m, \log(1/\delta)/\delta))$ for SSPPR-R (improving by m/n or $\log(1/\delta)$). These findings substantially strengthen the theoretical foundations of PPR computation: For SSPPR-R, our new upper and lower bounds coincide for graphs with $m \in \Omega(n\log^2(n))$ and any threshold $\delta, 1/\delta \in O(\text{poly}(n))$, indicating that we have achieved *theoretical optimality* under most graph regimes; the SSPPR-A query attains partial optimality under rough absolute error requirements (i.e., larger ϵ), where the upper bound simplifies to $O(1/\epsilon^2)$. This also matches our new lower bound result. To the best of our knowledge, this is the first work to establish an *optimal algorithmic result* for SSPPR queries. Further, our proposed techniques and results are generalizable to other relevant topics. Our lower-bound framework naturally extends to the *Single-Target Personalized PageRank* (STPPR) query, improving its lower bound from $\Omega(\min(n, 1/\delta))$ to $\Omega(\min(m, \frac{n}{\delta}\cdot\log n))$ under standard arc-centric model. This new result also matches the upper bound established in [36], revealing the *optimality* and underscoring its theoretical versatility.

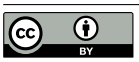
CCS Concepts: • **Theory of computation → Data structures and algorithms for data management**; **Proof complexity**; • **Mathematics of computing → Probabilistic algorithms**.

Additional Key Words and Phrases: Personalized PageRank; Upper Bound; Lower Bound;

ACM Reference Format:
Xinpeng Jiang, Haoyu Liu, Siqiang Luo, and Xiaokui Xiao. 2026. Near-Optimality for Single-Source Personalized PageRank. *Proc. ACM Manag. Data* 4, 2 (PODS), Article 110 (May 2026), 21 pages. https://doi.org/10.1145/3801906

*Full version available at [16]. Authors are listed in alphabetical order. Siqiang Luo is the corresponding author.

Authors' Contact Information: Xinpeng Jiang, College of Computing and Data Science, Nanyang Technological University, Singapore, xinpeng006@e.ntu; Haoyu Liu, College of Computing and Data Science, Nanyang Technological University, Singapore, haoyu.liu@ntu.edu.sg; Siqiang Luo, College of Computing and Data Science, Nanyang Technological University, Singapore, siqiang.luo@ntu.edu.sg; Xiaokui Xiao, School of Computing, National University of Singapore, Singapore, xkxiao@nus.edu.sg.



ACM 2836-6573/2026/5-ART110
https://doi.org/10.1145/3801906

1 Introduction

The *Single-Source Personalized PageRank* (SSPPR) query, first proposed in [12], has become a fundamental graph operation, attracting sustained interest from both academia and industry. Given a graph G, the SSPPR vector $\pi(s,\cdot)$ with respect to a source node s—PPR values of all nodes from s, is defined as the probability of an α—decay Random Walk (α-RW), with probability α to stop at the current node and probability $1-\alpha$ to randomly transition to neighbors, starting from s and terminating at any node t as $\pi(s,t)$. This definition naturally captures the importance of any node t relative to s, giving rise to widely used applications such as web search [4, 45], recommendation systems [5, 24, 31, 46], graph sparsification [33, 47], and graph neural networks [3, 17, 22, 23, 34].

Exact computation of all SSPPR vectors is expensive, requiring matrix inversion with $\Omega(n^2 \log n)$ time [32]. Consequently, considerable effort has focused on approximation algorithms for SSPPR query to balance efficiency with error guarantees. As such, a series of approximation algorithms have emerged [1, 6, 18, 20, 25, 29, 40, 44, 48], focusing on improved computational upper bounds based on two error settings: *absolute error* (SSPPR-A), which requires $|\hat{\pi}(s,t)-\pi(s,t)| \leq \epsilon$ for all t, and *relative error* (SSPPR-R), which requires $|\hat{\pi}(s,t)-\pi(s,t)| \leq c \cdot \pi(s,t)$ for all t with $\pi(s,t) \geq \delta$. Among the literature, MC [6], FORA [39] and SpeedPPR [43] together give the best known upper bound results of $O\left(\min(\log(1/\epsilon)/\epsilon^2, \sqrt{m\log n}/\epsilon, m\log(1/\epsilon))\right)$ for SSPPR-A and $O\left(\min(\log(1/\delta)/\delta, \sqrt{m\log(n/\delta)}, m\log(\log(n)/(\delta m)))\right)$ for SSPPR-R query. Though large amount of subsequent works [13, 14, 19, 21, 24, 38] try to further improve the computing overhead, such theoretical upper bound has been struck for nearly half decade. Despite the vitality in upper bounds, the lower bounds of SSPPR query remains far less developed. The best known lower bounds are $\Omega(\min(n, 1/\varepsilon))$ for SSPPR-A with ϵ absolute error [42] and $\Omega(\min(n, 1/\delta))$ for SSPPR-R with a constant relative error, both largely derived from worst-case output-size arguments or from the simpler *single-pair* PPR setting, due to the difficulties of constructing hard instances that force *any* algorithm to incur substantial computation on all nodes to meet the error requirement. This leaves a substantial theoretical gap yet to be closed.

In this work, we largely close the gap between upper-bounds and lower-bounds in SSPPR queries, as summarized in Table 1. On the **upper-bound** side, we propose two new algorithms, **RoundWalks** and **DistWalks**, which together tighten the computational complexities of the SSPPR-A and SSPPR-R queries. These results improve upon the best-known bounds by factors of $\log(1/\epsilon)$ and $\log(\log(n)/(m\delta))$. On the **lower-bound** side, we establish significantly stronger results: $\Omega(\min(m, 1/\epsilon^2))$ for SSPPR-A (improving by m/n or $1/\epsilon$) and $\Omega(\min(m, \log(1/\delta)/\delta))$ for SSPPR-R (improving by m/n or $\log(1/\delta)$). We highlight that, for SSPPR-R, our upper and lower bounds coincide for most graphs with $m \in \Omega(n\log^2(n))$ and any threshold $\delta, 1/\delta \in O(\text{poly}(n))$, indicating that our algorithm **DistWalks** achieves *theoretical optimality* under most graph regimes. Besides, the SSPPR-A query attains partial optimality under rough absolute error requirements (i.e., larger ϵ), where the upper bound simplifies to $O(1/\epsilon^2)$ via **RoundWalks**, matching our new lower bound result. To the best of our knowledge, this is the first work to establish an *optimal algorithmic result* for SSPPR queries, marking a significant advancement in the foundational study of graph analysis.

Our lower-bound results are based on constructing a non-trivial hard instance, and this technique has impact beyond the SSPPR topic. For example, we successfully extend our lower-bound framework to establish a much stronger result for *Single-Target Personalized PageRank* (STPPR) query, improving its lower bound from $\Omega(\min(n, 1/\delta))$ to $\Omega(\min(m, \frac{n}{\delta}\cdot\log(n)))$ under standard arc-centric model. This new result also matches the computational upper bound established in [36], thereby revealing the *optimality* of our analysis and underscoring its theoretical robustness and versatility.

1.1 Problem Formulation

We focus on two error settings: relative and additive error guarantees, which are widely adopted in mainstream PPR studies [13, 19, 21, 24, 35, 37–39, 41, 44, 49, 50].

Definition 1.1. SSPPR with Relative Error Guarantee (SSPPR-R). Given a graph G, a decay factor α, a source node s, a relative error parameter $c \in (0, 0.5)$, an overall failure probability parameter p_f and choosing a targeting PPR threshold $\delta \in (0, 1)$, SSPPR-R query requires an estimated SSPPR vector $\hat{\pi}(s, \cdot)$ such that with at least $1 - p_f$ probability, $|\hat{\pi}(s, t) - \pi(s, t)| \leq c \cdot \pi(s, t)$ holds for all node t with its PPR value $\pi(s, t) \geq \delta$. Note that the threshold δ can be chosen arbitrary close to 0 or 1, which correspond to the shallow or large targeting PPR values. Following conventions in prior work [15, 24, 26, 37], parameters α, c, and p_f are treated as constants.

Definition 1.2. SSPPR with Additive Error Guarantee (SSPPR-A). Given a graph G, a decay factor α, a source node s, an additive error parameter $\varepsilon \in (0, 1)$ and an overall failure probability parameter p_f, SSPPR-A query requires an estimated SSPPR vector $\hat{\pi}(s, \cdot)$ such that with at least $1 - p_f$ probability, $|\hat{\pi}(s, t) - \pi(s, t)| \leq \varepsilon$ holds for all node t.

Definition 1.3. Single Target Personalized PageRank Query (STPPR). Given a graph G, a decay factor α, a target node s, a relative error parameter $c \in (0, 0.5)$, an overall failure probability parameter p_f and choosing a targeting PPR threshold $\delta \in (0, 1)$, STPPR query requires an estimated STPPR vector $\hat{\pi}(s, \cdot)$ such that with at least $1 - p_f$ probability, $|\hat{\pi}(s, t) - \pi(s, t)| \leq c \cdot \pi(s, t)$ holds for all node s with its PPR value $\pi(s, t) \geq \delta$.

1.2 Existing Results for SSPPR-A, SSPPR-R and STPPR

Upper Bounds. Prior arts primarily use three foundational algorithmic paradigms, namely, Monte Carlo (MC), Forward Push (FP), and Backward Push (BP), and have been further refined through various state-of-the-art (SOTA) techniques. The MC method provides a sample-based approximation to the SSPPR vector using α-decay random walks, and achieves high-probability relative error guarantees with expected time complexity $O(\log(n)/\delta)$ for SSPPR-R, or $O(\log(1/\varepsilon)/\varepsilon^2)$ for SSPPR-A. The FP method is a local-push algorithm that simulates random walks deterministically via graph traversal. To guarantee an additive error of ε, it requires setting a residue threshold $r_{\max} = \varepsilon/m$, resulting in a worst-case time complexity of $O(m/\varepsilon)$. The BP method is similarly structured but performs the push operation in reverse, tailored to STPPR. With *global* BP, $O(m \log(1/\delta))$ is required for solving STPPR. BP can also be applied naively to SSPPR-A for all target nodes with $O(m/\varepsilon)$ complexity. To overcome the inefficiencies of basic PPR algorithms, hybrid methods that combine Monte Carlo (MC) sampling with local-push strategies have been developed. Wang et al. [40] improves upon MC by first using FP and then refining the result via targeted MC. Balancing these two phases, it achieves an expected complexity of $O\big(\sqrt{(m \log n)/\delta}\big)$ for SSPPR-R query. Wu et al. [44] restructures FP to enforce a more regular push schedule, reducing total residual mass more efficiently and yielding $O\big(m \log(1/\varepsilon)\big)$ for SSPPR-A and comparable efficiency for SSPPR-R when combined with MC. More recently, Wei et al. [42] integrates MC with an adaptive variant of Backward Push (BP): it first constructs a candidate set through rough MC estimation and then uses selective backward pushes and extra walks for refinement, attaining sublinear expected complexity $O\big(\sqrt{m \log n}/\varepsilon\big)$ for SSPPR-A. For the single-target case, Wang et al. [36] injects randomness into BP's deterministic push, maintaining an unbiased estimator with bounded variance while skipping negligible probability mass, thus answering STPPR in $O\big(\frac{n}{\delta} \cdot \log(n)\big)$ time.

Lower Bounds. Unlike the extensive body of work on improving *upper* bounds for SSPPR computation, rigorous *lower* bounds remain sparse and often heuristic, offering limited insight into the fundamental complexity of the problem. The earliest bound relevant to Personalized PageRank (PPR) is due to Lofgren et al. [27], which gives a lower bound of $\Omega(\sqrt{1/\delta})$ for *single-pair* PPR with relative error. However, this is dominated by the trivial output-size bound $\Omega\big(\min(n, 1/\delta)\big)$ obtained by noting that there can be up to $\min(n, 1/\delta)$ targets t with $\pi(s, t) \geq \delta$. To the best of our knowledge,

no other nontrivial *direct* lower bounds are known for SSPPR-R. For the absolute-error variant (SSPPR-A), Wei et al. [42] proves $\Omega\big(\min(n, 1/\varepsilon)\big)$ by the same output-size reasoning: any algorithm must return nonzero estimates for all t with $\pi(s,t) \geq \varepsilon$, of which there may be $\Omega\big(\min(n, 1/\varepsilon)\big)$. While conceptually simple, such arguments understate the difficulty of producing accurate estimates for *all* nodes within prescribed error margins. Likewise, STPPR query, Wang et al. [36] shows lower bounds of $\Omega(n)$ in the worst case and $\Omega\big(\min(n, 1/\delta)\big)$ on average, which still leaves a substantial gap relative to known upper bounds unlike other problems [28].

1.3 Our Results and Discussion

In this section, we present our main results summarized in Table 1. All our arguments are developed under the standard *arc-centric model [7–11]*, which will be formally defined in Section 2.

Table 1. Best-Known results and our Improvements of SSPPR-A, SSPPR-R and STPPR query. We highlight our new results in blue and bold the **Improvements** when error parameters ϵ, δ are sufficiently small. For simplicity, we denote $F_A(m,n,\epsilon) := \min(\frac{1}{\epsilon}\sqrt{m\log(n)}$ [42], $m\log\frac{1}{\epsilon}$ [44]).

Query	**Best-Known Lower Bounds**	**New Lower Bounds**[a]	**Improvements**
SSPPR-R	$\Omega\left(\min(n, \frac{1}{\delta})\right)$	$\Omega\left(\min(m, \frac{\log(\frac{1}{\delta})}{\delta})\right)$	$\log(\frac{1}{\delta})$ or $\frac{\mathbf{m}}{\mathbf{n}}$
SSPPR-A	$\Omega\left(\min(n, \frac{1}{\epsilon})\right)$ [42]	$\Omega\left(\min(m, \frac{1}{\epsilon^2})\right)$	$\frac{1}{\epsilon}$ or $\frac{\mathbf{m}}{\mathbf{n}}$
STPPR	$\Omega\left(\min(n, \frac{1}{\delta})\right)$ [36]	$\Omega\left(\min(m, \frac{n}{\delta}\cdot\log(n))\right)$	$n\log(n)$ or $\frac{\mathbf{m}}{\mathbf{n}}$
Query	**Best-Known Upper Bounds**	**New Upper Bounds**	**Improvements**
SSPPR-R	$O\left(\min\left(\frac{\log(n)}{\delta}\right.\right.$[6], $\sqrt{\frac{m\log(n)}{\delta}}$[40], $\left.\left.m\log(\frac{\log(n)}{\delta m})[44]\right)\right)$	$O\left(\min\left(\frac{\log(\frac{1}{\delta})}{\delta},\right.\right.$ $\left.\left.m + n\log(n)\log(\frac{\log(n)}{\delta m})\right)\right)$	$\frac{\log(n)}{\log(\frac{1}{\delta})}$ or $\approx \log(\frac{\log(n)}{\delta m})$
SSPPR-A	$O\left(\min(\frac{\log(1/\epsilon)}{\epsilon^2}[6], F_A(m,n,\epsilon))\right)$	$O\left(\min\left(\frac{1}{\epsilon^2}, F_A(m,n,\epsilon)\right)\right)$	$\log(\frac{1}{\epsilon})$
STPPR	$O\left(\min(m\log(\frac{1}{\delta}), \frac{n}{\delta}\cdot\log(n))\right)$	–	–
Query	**Optimality**		
SSPPR-R	**DistWalks** [Ours]	any small $\delta, 1/\delta \in O(\text{poly}(n))$ and $m \in \Omega(n\log^2(n))$	
SSPPR-A	**RoundWalks** [Ours]	any small $\epsilon, 1/\epsilon \in O(\sqrt{m})$	
STPPR	RBS [36]	any small $\delta, 1/\delta \in O(\frac{m}{n\log(n)})$	

[a] A concurrent work [2] establishes lower bounds of $\Omega(\min(m, 1/\delta))$ for SSPPR (via a direct application of their Single-Node PPR result) and $\Omega(\min(m, n/\delta))$ for STPPR. Developed independently using different techniques, our results further surpass these bounds by introducing additional $\log(1/\delta)$ and $\log(n)$ factors, respectively.

Theorem 1.4 [Improved upper bound of SSPPR-R]. *We present **DistWalks** (Algorithm 3), which solves the SSPPR-R query within $O\left(m + n\log(n)\log(\frac{\log(n)}{m\delta})\right)$ queries in expectation.*

DistWalks fundamentally overcomes the limitations of prior state-of-the-art methods that rely on deterministic push operations for initialization. Instead, it introduces a novel decomposition of PPR by accumulating contributions from already discovered effective paths, thereby avoiding redundant random walks over previously visited nodes. This refined design achieves, for the first time, linear dependency on the graph size (m) in relatively dense graphs. This significantly tightens the upper-bound widely.

Theorem 1.5 [Improved upper bound of SSPPR]. *We present **RoundWalks** (Algorithm 1), which solves the SSPPR-A (SSPPR-R) query within $O(1/\epsilon^2)$ ($O(\log(1/\delta)/\delta)$) queries in expectation.*

The proposed Algorithm 1 employs a two-phase estimator, i.e., *discovery* and *estimation*, to improve upon naive Monte Carlo (MC) methods. This design achieves $\log(1/\epsilon)$ improvement of SSPPR-A.

Theorem 1.6 [Improved lower bound of SSPPR-R]. *Choose any decay factor $\alpha \in (0,1)$, failure probability $p \in (0,1)$, error parameter $c \in (0, \frac{1}{2}]$, and any functions $\delta(n) \in (0,1), m(n) \in \Omega(n) \cap O(n^2)$. Consider any oracle (possibly randomized) algorithm within the arc-centric model that, for an arbitrary graph G with $O(m)$ edges and $O(n)$ nodes and any source node s, with probability at least $1-p$ the oracle algorithm gives an estimation $\hat{\pi}(s,\cdot)$ for SSPPR vector $\pi(s,\cdot)$, satisfying $|\hat{\pi}(s,v) - \pi(s,v)| \le c \cdot \pi(s,v)$ for each node v at $\pi(s,v) \ge \delta$. Then there exists a graph* U *and a source node s such that:*

(i). U *contains $O(m)$ edges and $O(n)$ nodes;*

(ii). the algorithm requires $\Omega\left(\min(m, \frac{\log(1/\delta)}{\delta})\right)$ queries to succeed in expectation.

Theorem 1.6 improves the best-known lower bound from $\Omega\big(\min(n, 1/\delta)\big)$ to $\Omega\big(\min(m, \log(1/\delta)/\delta)\big)$. This improvement is substantial, adding a multiplicative factor of up to m/n, which can reach $\Theta(n)$ on dense graphs such as cliques when high approximation precision is required. Moreover, the entire proof framework is readily adaptable. By inverting all edge directions in the SSPPR-R construction, we derive Theorem 1.7 for the STPPR query:

Theorem 1.7 [Improved Lower Bound of STPPR Query]. *Choose any decay factor $\alpha \in (0,1)$, failure probability $p \in (0,1)$, error parameter $c \in (0, \frac{1}{2}]$, and any functions $\delta(n) \in (0,1), m(n) \in \Omega(n) \cap O(n^2)$. Consider any oracle (possibly randomized) algorithm within the arc-centric model that, for an arbitrary graph G with $O(m)$ edges and $O(n)$ nodes and any target node t, with probability at least $1-p$ the oracle algorithm outputs an estimation $\hat{\pi}(\cdot,t)$ for STPPR vector $\pi(\cdot,t)$, satisfying $|\hat{\pi}(v,t) - \pi(v,t)| \le c \cdot \pi(v,t)$ for each node v at $\pi(v,t) \ge \delta$. Then there exists a graph* U *and a target node t such that:*

(i). U *contains $O(m)$ edges and $O(n)$ nodes;*

(ii). the algorithm requires $\Omega\left(\min(m, \frac{n}{\delta} \cdot \log(n))\right)$ queries to succeed in the worst case.

Benefiting from our flexible proof framework, Theorem 1.7 significantly strengthens the best-known worst-case lower bound from $\Omega\big(\min(n, 1/\delta)\big)$ to $\Omega\big(\min(m, \frac{n}{\delta} \cdot \log(n))\big)$. Crucially, this lower bound matches the known computational upper bound in the prevalent setting $\delta = O(1/n)$, which is standard in PPR query computation [24, 26, 40, 43]. This implies that the computational complexity of STPPR query is essentially *optimal*, unveiling the maturity in PPR algorithm development.

Theorem 1.8 [Improved Lower Bound of SSPPR-A Query]. *Choose any decay factor $\alpha \in (0,1)$, failure probability $p \in (0,1)$, and any functions $\varepsilon(n) \in (0,1), m(n) \in \Omega(n) \cap O(n^2)$. Consider any oracle (possibly randomized) algorithm within the arc-centric model that, for an arbitrary graph G with $O(m)$ edges and $O(n)$ nodes and any source node s, with probability at least $1-p$ the oracle algorithm gives an estimation $\hat{\pi}(s,\cdot)$ for SSPPR vector $\pi(s,\cdot)$, satisfying $|\hat{\pi}(s,v) - \pi(s,v)| \le \varepsilon$ for each node v. Then there exists a graph* U *and a source node s such that:*

(i). U *contains* $O(m)$ *edges and* $O(n)$ *nodes;*

(ii). the algorithm requires $\Omega\left(\min(m, \frac{1}{\epsilon^2})\right)$ *queries to succeed in expectation.*

Theorem 1.8 strengthens the best-known lower bound from $\Omega(\min(n, 1/\epsilon))$ to $\Omega\big(\min(m, 1/\epsilon^2)\big)$. This improvement is substantial and, strikingly, the $\frac{1}{\epsilon^2}$ term partially matches our improved computational upper bound for the SSPPR-A query. This alignment indicates that our new algorithm (Algorithm 1) is essentially *optimal* for SSPPR-A when the error requirement is relatively coarse.

2 Notations and Tools

Notations. Given an underlying directed graph $G = (V_G, E_G)$, let $n = |V_G|$ and $m = |E_G|$ denote the number of nodes and edges, respectively. Each edge $e = (u, v) \in E_G$ indicates that node u is a parent of node v, and correspondingly, v is a child of u. For a node set V, we denote its elements either as $v \in V$ or using index notation $V[i]$ to refer to the node at index i. The out-degree and in-degree of a node u are denoted by $d_{\text{out}}(u)$ and $d_{\text{in}}(u)$, respectively. In particular, we denote by $\deg_V(v)$ the number of edges between node v and all potential nodes in the set V. A permutation of size n is defined as a bijection $\mathrm{p}^n \in \mathrm{S}^n : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$; equivalently, p^n is an element of the symmetric group S^n. For any index $0 \le i < n$, we use $\mathrm{p}[i]$ to represent the permuted index.

Definition 2.1 [multigraph and edge multiplicity [30]]. A *multigraph* is a graph that may contain parallel edges (multiple edges between the same pair of nodes); a *simple graph* contains no such parallel edges. The *edge multiplicity* of a multigraph U is defined as $\mathrm{MUL}(G) = \max_{u,v \in V} \big|\{ e \in E_{\mathrm{U}} \mid e = (u, v)\}\big|$, representing the maximum number of parallel edges between any node pair.

Definition 2.2 [Arc–Centric Graph Query Model [10, 11]]. We focus on the query complexity under the standard RAM model. To formalize graph interactions, we allow algorithms to access the underlying graph G under the standard *arc-centric graph-access model* [10, 11], where the oracle supports the following queries in unit time:

(i). JUMP(): returns a uniformly random node $v \in V$ and marks v as *covered*;

(ii). INDEG(v)/OUTDEG(v): for any *covered* node $v \in V$, returns its in/out-degree $d_{in}(v)/d_{out}(v)$;

(iii). $\mathrm{ADJ}_{in}(v, i)/\mathrm{ADJ}_{out}(v, i)$: for any *covered* node $v \in V$ and integer $1 \le i \le d_{in}(v)/d_{out}(v)$, returns the i^{th} in/out-neighbor of node v.

Definition 2.3 [Graph with edge and node indices]. Following the notation in [10], we refer to a graph $G = (V, E)$ with edge and node indices with the following labels: (i). Node labels. Each node $v \in V$ is assigned a unique label from the set $\{1, \ldots, |V|\}$; (ii). Edge labels. For each node $v \in V$, its in-edges and out-edges are labeled with integers of $\{1, 2, \ldots, d_{\text{in}}(v)\}$ and $\{1, 2, \ldots, d_{\text{out}}(v)\}$.

Remark. Due to space limitations, we defer all proof details to the online full version [16]. In particular, detailed proofs for lemmas 3.1 to 3.4, 4.1, 4.2, 4.5, 4.6 and 5.1 to 5.3 are provided there.

3 Improved Upper Bounds for SSPPR Queries

In this section, we present our advances on the upper bounds for approximating SSPPR queries. Existing upper bounds do *not* imply any optimality, even when combined with our new lower bounds in Table 1. Specifically, the best-known SSPPR-R upper bound $F_R = O\Big(\min\Big(\frac{\log(n)}{\delta}, \sqrt{\frac{m\log(n)}{\delta}}, m\log\frac{\log(n)}{\delta m}\Big)\Big)$ reveals two fundamental gaps from the lower bound $f_R = \Omega\Big(\min\Big(\frac{\log(1/\delta)}{\delta}, m\Big)\Big)$: for moderate thresholds (e.g., $\delta = 1/\log(n)$) a persistent $\log(n)$ gap remains, and for very small δ, an extra $\log(\frac{\log(n)}{\delta m})$ factor prevents matching the $\Omega(m)$ bound. To close these two gaps, we introduce two complementary algorithms: **RoundWalks**, which refines the classical MC estimator to eliminate the unnecessary global $\log(n)$ dependence and achieves $O(\log(1/\delta)/\delta)$ for moderate-δ regimes; and **DistWalks**, which leverages new insights into state-of-the-art SSPPR techniques and a novel path-based PPR

Algorithm 1: RoundWalks $(G, s, \alpha, \delta, p_f)$

Input: Graph G, source $s \in V$, decay factor α, threshold δ, failure probability p_f.
Output: $V_{\text{out}} \subseteq V$.

1 $T_1 \leftarrow \frac{1}{\delta}\left(\log\frac{1}{\delta} + \log\frac{2}{p_f}\right)$; $T_2 \leftarrow \frac{20}{\delta}\left(\log\frac{1}{\delta} + \log\frac{2}{p_f}\right)$; $V_{\text{out}}^{\text{raw}} \leftarrow \varnothing$; $H(\cdot) \leftarrow \mathbf{0}$;
2 **for** $i \leftarrow 1$ **to** T_1 **do**
3 $v \leftarrow$ end of an α-RW from s;
4 $V_{\text{out}}^{\text{raw}} \leftarrow V_{\text{out}}^{\text{raw}} \cup \{v\}$;
5 $\theta \leftarrow 10\left(\log\frac{1}{\delta} + \log\frac{2}{p_f}\right)$; # For refining the returned node set.
6 **for** $i \leftarrow 1$ **to** T_2 **do**
7 $v \leftarrow$ end of an α-RW from s;
8 $H[v] \leftarrow H[v] + 1$;
9 $V_{\text{out}} \leftarrow \{u \in V_{\text{out}}^{\text{raw}} \mid H[u] \geq \theta\}$;
10 **return** V_{out}

decomposition to achieve $O\left(m + n\log(n)\log(\frac{\log(n)}{\delta m})\right)$, matching the $\Omega(m)$ lower bound on sufficiently dense graphs (e.g., $m \in \Omega(n\log^2 n)$). Together, these results yield an improved overall upper bound $O\left(\min\left(\frac{\log(1/\delta)}{\delta}, m + n\log(n)\log(\frac{\log(n)}{\delta m})\right)\right)$, matching the lower bounds for most practical settings where $1/\delta \in \text{poly}(n)$ and m is moderately large. We first introduce the main idea of **RoundWalks**.

Improve the MC Estimator. The classical MC estimator [6] enforces a *uniform* accuracy guarantee over all n nodes, which introduces an unavoidable $\log(n)$ factor via a union bound (appearing as $\frac{\log(n)}{\delta}$). This is well matched to regimes with *extreme* precision (e.g., $\delta \approx 1/n$) but overly conservative for *moderate* ones (e.g., $\delta \approx 1/\log(n)$). Such a global $\log(n)$ term becomes a bottleneck for overall approximation. Motivated by this, our idea is, we can first conduct a *modest* number of random walks to obtain an initial estimate that, with high probability, contains all required target nodes—i.e., a node set R_{out}—and then we restrict high-precision estimation only to nodes in R_{out} via another batch of random walks. This way, we can replace the global $\log(n)$ term with a $\log(1/\delta)$ dependence governed by the candidate size. Accordingly, we present **RoundWalks** (Algorithm 1), which consists of the following two phases:

- Discovery (lines 1-4). We first conduct T_1 α-RWs to find an initial candidate node set V_{out} such that, with high probability, all nodes of interest $V_{\geq\delta} := \{u : \pi(s,u) \geq \delta \mid u \in V\}$ are included.
- Estimation (lines 5-9). We then restrict high-precision estimation to V_{out} by running T_2 additional walks. By filtering nodes with tiny values, we'll ensure that the final candidate set V_{out} contains only meaningful nodes that $V_{\text{out}} \subseteq V_{\geq\delta/(5e)} := \{v : \pi(s,v) \geq \delta/(5e)\}$.

By rigorously setting T_1 and T_2, we conclude the following approximation guarantee in Lemma 3.1.

Lemma 3.1. *With probability* $\geq 1 - p_f$*, the node set* V_{out} *returned by Algorithm 1 satisfies:*

$$V_{\geq\delta} \subseteq V_{\text{out}} \subseteq V_{\geq\delta/(5e)}.$$

Based on this discovered candidate set from Algorithm 1, we can then estimate SSPPR values only for nodes in R_{out} using the standard MC. For SSPPR-A query, we further run

$$T = 256[\log\left(\tfrac{1}{\delta/(5e)}\right) + \log\left(\tfrac{2}{p}\right)]/[c^2\,(\delta/(5e))] = O(\log(1/\delta)/\delta)$$

independent α-RWs and for each $u \in V_{\text{out}}$ estimate $\pi(s,u)$ by empirical frequency, we then answer SSPPR-R query with overall failure probability $\leq p_f$ combining the Chernoff and union bound.

For SSPPR-A query, we then replace the error threshold δ with ϵ. By Lemma 3.1, with probability $\geq 1 - p_f/2$ we obtain $V_{\geq\varepsilon} \subset V_{\text{out}} \subset V_{\geq\varepsilon/(5e)}$. Again, we execute $T = 1024\log\left(\frac{2}{p_f}\right)/\left(\frac{\varepsilon}{5e}\right)^2 = O\left(\frac{1}{\varepsilon^2}\right)$ independent α-RWs from s, and estimate $\pi(s,u)$ for $u \in V_{\text{out}}$ by the empirical frequency with absolute tolerance $\varepsilon/(5e)$. Since $\pi(s,u) \geq \varepsilon/(5e)$ for all $u \in V_{\text{out}}$, the Chernoff bound yields

$$\mathbb{P}[\text{fail at } u] \leq 2\exp\left(-\tfrac{1}{3}\,\pi(s,u)\left(\frac{\varepsilon/(5e)}{\pi(s,u)}\right)^2 T\right) \leq 2\exp\left(-\ln\tfrac{2}{p} - \frac{1}{\pi(s,u)}\right) \leq \frac{p_f}{2}\cdot\pi(s,u).$$

Summing over u with $\sum_u \pi(s,u) = 1$ gives total estimation failure at most $p_f/2$. Together with the probability in discovery phase, the overall failure probability is bounded at most p_f.

Combining these steps, we finally achieve optimized complexities: SSPPR-A at $O(1/\varepsilon^2)$ and SSPPR-R at $O(\log(1/\delta)/\delta)$, supporting the main claims in Theorem 1.5.

3.1 DistWalks

Building on **RoundWalks**, we observe that the MC approach is highly effective for producing an initial, relatively accurate estimate. Leveraging this insight, we introduce **DistWalks**, which combines **RoundWalks** with a new decomposition of SSPPR computation based on paths. We first review the bottlenecks of existing techniques, to motivate our idea.

Current Bottlenecks. Current SOTA methods (e.g., FORA and SpeedPPR) combine push and walk procedures based on a key *invariant* decomposition of PPR: $\pi(s,t) = \hat{\pi}(s,t) + \sum_{u\in V} r(u)\pi(u,t)$, where $\hat{\pi}(s,\cdot)$ is the current estimation vector and $r(\cdot)$ denotes the residual values, which is based on the iterative definition of PPR vectors of $\pi(s,\cdot) = (1-\alpha)\mathbf{P}\pi(s,\cdot) + \alpha\mathbf{1}_s$. These methods typically perform push operations to initialize $\hat{\pi}(s,\cdot)$, followed by α-random walks to estimate the residual term using $r(u)$ on active nodes. While this hybrid design achieves notable improvements over MC, we identify two critical bottlenecks:

(i) The deterministic push phase expends substantial computation on *heavy* nodes with large $\pi(s,v)$, even though their scores could be efficiently estimated with only a few random walks.
(ii) The walk phase often revisits nodes whose estimates are already accurate, resulting in redundant computations and poor reuse of previously gathered information for refinement.

These inefficiencies originate from the traditional decomposition itself, which overlooks the deeper relationship between *PPR and its underlying path summations*. Motivated by this insight, we propose a novel *path-based decomposition* that yields more accurate initial estimates while fully reusing observed information, eliminating redundant α-RW steps and reducing computational complexity.

A New Path-based Decomposition. Let $\overleftrightarrow{p_G}(u,v)$ denote the set of all directed paths between any node pair (u,v) in graph G. For any path $\overleftrightarrow{p} \in \overleftrightarrow{p_G}(u,v)$, we define its weight $w_G(\overleftrightarrow{p})$ as the probability that an α-decay random walk (RW) traverses $\overleftrightarrow{p}$. For example, for a single-edge path $\overleftrightarrow{p_1} = \{u,v\}$ with $(u,v) \in E$, we have $w_G(\overleftrightarrow{p_1}) = \frac{1-\alpha}{d_{\text{out}}(u)}$. For two paths $\overleftrightarrow{p_1} \in \overleftrightarrow{p_G}(u_2,u_3)$ and $\overleftrightarrow{p_2} \in \overleftrightarrow{p_G}(u_1,u_2)$, their concatenation $\overleftrightarrow{p_1}\circ\overleftrightarrow{p_2} \in \overleftrightarrow{p_G}(u_1,u_3)$ satisfies $w_G(\overleftrightarrow{p_1}\circ\overleftrightarrow{p_2}) = w_G(\overleftrightarrow{p_1})\,w_G(\overleftrightarrow{p_2})$. Without loss of generality, we assume the graph contains no dangling nodes and define the weight of zero-length paths to be zero. Under this formulation, the PPR can be equivalently expressed as $\pi(s,t) = \alpha\sum_{\overleftrightarrow{p}\in\overleftrightarrow{p_G}(s,t)} w_G(\overleftrightarrow{p})$, where the summation aggregates contributions from all feasible paths between s and t. Building upon this, we then introduce a new path-based *decomposition* that focuses on estimating PPR values of any subset of nodes. This key component plays a fundamental role in our final estimator for SSPPR-R query, replacing the traditional Push-and-Walk paradigm while allowing *sufficient re-using* of intermediate estimations and avoid re-visiting redundant nodes.

Specifically, given any subset $X \subset V$ of nodes with source node $s \notin X$, we define the *boundary* set $\textcircled{X}$, to contain all nodes in $V\backslash X$ which have an out-edge to node set X. As such, any path $\overleftrightarrow{p}$ starting from s and ending at nodes in X can be linked as $\overleftrightarrow{p_1}\circ\overleftrightarrow{p_2}$, where $\overleftrightarrow{p_1}$ is a path ending in $\textcircled{X}$

Algorithm 2: RoundEst $(G, X, \hat{\pi}_G(s,\cdot), \alpha, c_r, \delta_r, p_r)$

Input: Graph G; target set X; initial estimation $\hat{\pi}_G(s,\cdot)$; decay factor α; error parameter c_r; threshold δ_r; failure probability p_r.

Output: Detected set $X_{out} \in X$ and updated estimation $\hat{\pi}_G(s,\cdot)$.

Construct G_X following steps (A).(B).(C).

$\forall u \in \textcircled{X}, \hat{\mathbf{S}}_\mathbf{r}(u) \leftarrow d^{G_X}_{\text{out}}(u)\hat{\pi}_G(s,u)/(\alpha\, d^{G}_{\text{out}}(u))$; $\mathtt{SUM}(\hat{\mathbf{S}}_\mathbf{r}) \leftarrow \sum_{u\in\textcircled{X}} \hat{\mathbf{S}}_\mathbf{r}(u), \hat{\hat{\mathbf{S}}}_\mathbf{r} \leftarrow \hat{\mathbf{S}}_\mathbf{r}/\mathtt{SUM}(\mathbf{S_r})$;

Construct an alias table for $\hat{\hat{\mathbf{S}}}_\mathbf{r}$;

Use it to execute $X_{out} \leftarrow$ **RoundWalks**$(G_X, \hat{\hat{\mathbf{S}}}, \alpha, \delta_r, p_r)$;

$N_r \leftarrow 256\alpha \log(n/p_r)/\left(c_r^2\delta_r\right), H(\cdot) \leftarrow \mathbf{0}$;

for $i = 1, 2, \ldots, N_r$ **do**

  Draw a node $u \sim \hat{\hat{\mathbf{S}}}$;

  Let $v \leftarrow$ end of i-th α-RW from u on G_X;

  if $v \in X_{out}$ **then** $H[v] \leftarrow H[v] + \frac{1}{N_r}$;

$\hat{\pi}_G(s,t) \leftarrow H(t) \cdot \mathtt{SUM}(\hat{\mathbf{S}}_\mathbf{r})$ for $\forall t \in X_{out}$;

return X_{out}, $\hat{\pi}_G(s,\cdot)$

and $\overleftrightarrow{p_2}$ starts in $\textcircled{X}$, has its first edge entering X, and then stays entirely within X (or intuitively, it is the path ever since the last visit of outside X). This partition exists and is unique for each $\overleftrightarrow{p} \in \overleftrightarrow{p_G}(s,u)$ for any source s and node $u \in V$. Thereby, for any node $t \in X$ we have that

$$\pi(s,t) = \alpha \sum_{u\in\textcircled{X}} \sum_{\overleftrightarrow{p_1}\in\overleftrightarrow{p_G}(s,u)} w_G(\overleftrightarrow{p_1}) \sum_{\overleftrightarrow{p_2}\in\overleftrightarrow{p_G}(u,t|X)} w_G(\overleftrightarrow{p_2}) \tag{1}$$

$$= \sum_{u\in\textcircled{X}} \pi(s,u) \sum_{\overleftrightarrow{p_2}\in\overleftrightarrow{p_G}(u,t|X)} w_G(\overleftrightarrow{p_2}), \tag{2}$$

where $\overleftrightarrow{p_G}(u,t|X)$ denotes paths that only contain nodes in X except the starting point. Equation 1 indicates that $\pi(s,t)$ can be expressed as the combination of: (i) intermediate contributions from the boundary set via $\pi(s,u)$ for $u \in \textcircled{X}$, and (ii) a restricted path–weight aggregation over paths that stay entirely inside X after entering it. Consequently, once accurate estimates of $\pi(s,u)$ are available for all $u \in \textcircled{X}$, the remaining contribution to $\pi(s,t)$ for $t \in X$ can be computed using only nodes and edges within X, without revisiting any node outside X. A key observation is that the residual term $\sum_{p_2\in\overleftrightarrow{p_G}(u,t|X)} w_G(p_2)$ can be equivalently expressed through PPR values on a carefully constructed auxiliary graph G_X. This equivalence is formalized below:

Lemma 3.2 [Decomposition]. *Given graph G, subset X and source node $s \notin X$, we have*

$$\pi_G(s,t) = \frac{1}{\alpha} \sum_{u\in\textcircled{X}} \pi_G(s,u)\pi_{G_X}(u,t) d^{G_X}_{out}(u)/d^{G}_{out}(u), \tag{3}$$

where G_X is constructed by:

(A). $V_{G_X} = X \cup \textcircled{X} \cup \{v_X\}$, where v_X is a new auxiliary node.

(B). For nodes $u \in V$ and $t \in X$, if directed edge $(u,t) \in E$, add this edge to E_{G_X}.

(C). For nodes $t \in X$ and $u \in V/X$, if directed edge $(t,u) \in E$, add edge (t, v_X) to E_{G_X}.

Intuitively, Step (A). eliminates any new dangling nodes introduced by the out-edge deletion, and Step (B). preserves all edges entering X, while Step (C). redirects all edges leaving X to the sink-like node v_X, ensuring that $d^{G_X}_{\text{out}}(t) = d^{G}_{\text{out}}(t)$ for all $t \in X$. This construction precisely encodes the restricted path aggregation in Equation 1. We emphasize that G_X is a conceptual object used for analysis and we avoid materialize it explicitly to maintain a bounded complexity later.

By Lemma 3.2, we can now construct an on-hand estimator to solve PPR estimations for any interested nodes in the given subset X. Based on an initially accurate estimation $\{\hat{\pi}_G(s,u) \mid u \in \textcircled{X}\}$ (e.g., by running MC), the remaining part becomes α-RW simulation for $\pi_{G_X}(u,\cdot)$ on graph G_X with distributed starting nodes $u \in \textcircled{X}$ valued $\hat{\mathbf{S}}_\mathbf{r}(u) := \frac{1}{\alpha}\hat{\pi}_G(s,u)d^{G_X}_{\text{out}}(u)/d^{G}_{\text{out}}(u)$. This sourced $\alpha-$RW simulation can be readily achieved by first constructing an alias table with normalized probability distribution $\hat{\bar{\mathbf{S}}}_\mathbf{r} := \hat{\mathbf{S}}_\mathbf{r}/\texttt{SUM}(\hat{\mathbf{S}}_\mathbf{r})$, where $\texttt{SUM}(\hat{\mathbf{S}}_\mathbf{r}) = \sum_{u\in\textcircled{X}} \hat{\mathbf{S}}_\mathbf{r}(u)$. This step costs $O(|\textcircled{X}|)$ for construction and supports sampling in $O(1)$ time and then simulate a proper amount of N_r $\alpha-$RWs. This gives rise to our *intermediate* Algorithm 2 (**RoundEst**), which can accurately estimate PPR values of nodes in target set X larger than the pre-defined threshold δ_r (returned as node set X_{out}), based on some initial estimation on the boundary set $\textcircled{X}$. With proper parameter settings, we guarantee the quality of algorithm **RoundEst** with relative error c_r and failure probability p_r:

Lemma 3.3 [RoundEst Guarantee]. *Assume n is sufficiently large and initial estimation satisfies $\forall u \in \textcircled{X}, \hat{\pi}_G(s,u)/\pi_G(s,u) \in [1-c, 1+c]$ for constant $c \in (0, \frac{1}{2}]$. Then with probability $\geq 1-p_r$, the output set X_{out} and updated estimation $\hat{\pi}_G(s,\cdot)$ from Algorithm 2 satisfy:*

(i) $X_{\geq \delta_r \cdot \texttt{SUM}(\hat{\mathbf{S}}_\mathbf{r})} \subset X_{out}$. That is—$\forall x \in X$ with $\pi_G(s,x) \geq \delta_r \cdot \texttt{SUM}(\hat{\mathbf{S}}_\mathbf{r})$, it's included in set X_{out}.
(ii) $\forall t \in X_{out}, \hat{\pi}_G(s,t)/\pi_G(s,t) \in [(1-c)(1-c_r), (1+c)(1+c_r)]$ holds for relative error guarantee.

By Lemma 3.3, Algorithm 2 provides a basic mechanism for solving the SSPPR-R query: one may simply add an auxiliary source s' together with a directed edge (s,s') to G, and take the initial target set $X = V \setminus \{s'\}$, with the boundary consisting solely of the source node s, for which $G_X = G, \hat{\pi}_G(s',s') = \pi_G(s',s') = \alpha$ is known exactly. Setting the round parameters to $c_r = c$, $p_r = p$, and $\delta_r = \delta$ makes **RoundEst** degenerate into the raw **RoundWalks** procedure, which performs α-RWs directly from s similarly. However, simulating α-RWs reveals a key structural property: nodes with larger PPR values relative to s (*heavy* nodes) are encountered far more frequently and therefore require significantly fewer samples to obtain accurate estimates compared to nodes with very small PPR values (*light* nodes). This motivates us a progressive refinement strategy. Instead of estimating all nodes at once, we expand the estimation frontier gradually—from heavy nodes to increasingly lighter nodes—by decreasing the threshold round-by-round, eventually reaching the target δ. This multi-round refinement greatly integrates with core of **RoundEst** that estimated heavy nodes will not be re-visited in subsequent rounds (as we only utilize G_X), avoiding redundant costs. To implement the above refinement strategy, we introduce a shrinking factor $\Delta > 1$ that exponentially decreases the detection threshold from an initial value $\delta_0 \in \Theta(\frac{1}{n})$ down to the target threshold $\delta_R \texttt{SUM}(\hat{\mathbf{S}}_\mathbf{r}) \leq \delta$ over $R \in O\log_\Delta(1/(n\delta))$ rounds. This yields the main procedure of **DistWalks** framework (Algorithm 3) for solving the SSPPR-R query with great complexity.

We begin by identifying an initial heavy node set V_0 using **RoundWalks** (line 2), and estimating their PPR values accurately via α-RWs (line 3). The complement $X_0 = V \setminus V_0$ then becomes the initial target set for refinement. For each round $r = 1, \ldots, R-1$, we invoke the single-round estimator **RoundEst** on the current target set X_{r-1}, producing a newly resolved node set V_r with certified relative-error guarantees. The remaining unresolved nodes form the next target set $X_r = X_{r-1} \setminus V_r$. As the remaining PPR scores decreases geometrically by factor Δ across rounds, the algorithm progressively expands its estimation coverage from heavy nodes toward light nodes. After R rounds, all nodes with $\pi(s,x) \geq \delta$ will then be included ideally. Although chaining multiple refinement rounds may seem to introduce cumulative approximation error, we show that all such errors can be rigorously controlled. By carefully choosing the parameters δ_r, c_r, and p_r in each round (line 5), the overall estimator remains within desired SSPPR-R query guarantees, formally established as below:

Lemma 3.4 [DistWalks Guarantee]. *Algorithm 3 correctly solves the SSPPR-R query: it returns an estimated SSPPR vector $\hat{\pi}(s,\cdot)$ that with probability at least $1-p_f$, it holds $|\hat{\pi}(s,u)-\pi(s,u)| \le c\cdot\pi(s,u)$ for all node $u \in V$ with PPR value $\pi(s,u) \ge \delta$.*

3.2 Complexity analysis

We next analyze the overall complexity. Recall that we avoid explicitly constructing G_X during each round of estimation. We first describe the implementation details needed to derive the result.

Implicit Construction of G_X. Instead of explicitly materializing G_X in each round r (denoted as $G_X^{(r)}$), we observe that it suffices to maintain and update the out-neighbors of nodes newly estimated in that round (i.e., the node set V_r). Initially, we construct $G_X^{(0)}$ by augmenting the original graph G with a global auxiliary node v_X. Then, immediately after the node set V_r is estimated in round r, we update $G_X^{(r-1)}$ to $G_X^{(r)}$ by processing all edges incident to each node $v_{\text{new}} \in V_r$ using the following steps, while maintaining the updated out-degree $d_{out}^{G_X}$.

(i) For any incoming edge $e=(u,v_{\text{new}})$ where $u \in X_r$ (the remaining nodes to be estimated), we redirect e to point to v_X, i.e., update e to (u,v_X).
(ii) For any incoming edge $e=(u,v_{\text{new}})$ where $u \notin X_r$ (nodes already estimated), we remove e.
(iii) For any outgoing edge $e=(v_{\text{new}},u)$ where $u \notin X_r$ (nodes already estimated), we remove e.
(iv) If after steps (b) or (c) an already estimated node $u \notin X_r$ satisfies $d_{out}^{G_X}(u)=0$, we remove u.

These steps dynamically recover $G_X^{(r)}$ in each round without reconstructing it from scratch. Consequently, the update cost for each new node $v_{\text{new}} \in V_r$ is bounded by $O(d_{in}(v_{\text{new}})+d_{out}(v_{\text{new}}))$. Since $\bigcup_r V_r$ equals the initial target set X_0, the total cost of all updates is bounded by $O(m)$.

Finally, we note that the construction of G_X may introduce parallel edges, which can be handled directly by algorithm **RoundWalks**. Alternatively, parallel edges can be avoided by adding $O(n)$ auxiliary (padding) nodes $u'_1,\dots,u'_n$ and redirecting edges in FIFO order. This modification does not increase the number of nodes in recursive applications of the construction, since the same padding nodes can be reused. The remaining cost arises from α-RW sampling and alias-table construction. Each round performs $N_r=\Theta((\log|X_r|/p_r)/(c_r^2\delta_r))$ independent α-random walks and builds an $O(n)$ alias table. Since the term $\texttt{SUM}(\hat{\mathbf{S}}_\mathbf{r})\delta_r$ shrinks geometrically and our parameter settings satisfy

Algorithm 3: DistWalks$(G,s,\alpha,c,\delta,p_f,\Delta)$

Input: Graph G; source node s; decay factor α; error c; threshold δ; probability p_f; shrinking factor Δ.
Output: Estimation $\hat{\pi}_G(s,\cdot)$.

$\hat{\pi}_G(s,\cdot) \leftarrow \mathbf{0}$, $\delta_0 \leftarrow \frac{\alpha}{(1+c)n\Delta}$, $R \leftarrow \log_\Delta(\frac{2\delta_0}{\alpha\delta})$;
$V_0 \leftarrow$ **RoundWalks**(G,s,δ_0,p_f);
Cast $\frac{320R^2\log(n/p_f)}{c^2\delta_0}$ α-RWs from s to estimate $\hat{\pi}_G(s,u)$ for all $u \in V_0$;
$X_0 \leftarrow V\backslash V_0$;
for $r=1,\dots,R-1$ **do**
  $c_r=\frac{c}{4R}$, $\delta_r=\delta_0$, $p_r=\frac{p_f}{2n}$;
  $V_r,\hat{\pi}_G(s,\cdot) \leftarrow$ **RoundEst**$(G,G_X,X_{r-1},\hat{\pi}_G(s,\cdot),\alpha,c_r,\delta_r,p_r)$;
  $X_r \leftarrow X_{r-1}\backslash V_r$;
return $\hat{\pi}_G(s,\cdot)$

$\delta_r = \Omega(1/n)$ and $c_r = \Omega(1/R)$, the algorithm terminates in $O(\log(1/(n\delta)))$ rounds. This yields an overall complexity of $O\big(m + n \log n \log^3(1/(n\delta))\big)$ for **DistWalks**.

Reducing log **factors.** So far, we have presented the core framework of **DistWalks**, which establishes a computational upper bound of $O(m + n \log n \log^3(1/(n\delta)))$ for the SSPPR-R query. While this bound already demonstrates near-optimality in dense regimes, the logarithmic factor $\log^3(1/(n\delta))$ can be further improved to $\log(\frac{\log n}{m\delta})$ through additional algorithmic optimizations, which matches the bound stated in Theorem 1.4. To achieve this improvement, we introduce the following two technical refinements without altering the overall framework:

(i) **Adaptive thresholds for δ_r.** We employ a doubling strategy to dynamically determine the threshold δ_r in each round. This allows the computational cost to be amortized across rounds, scaling as $O(n/R)$ instead of the worst-case $O(n)$ per round.

(ii) **Global error analysis.** We revisit the accuracy guarantees from a global perspective. Instead of relying on worst-case error propagation from previous rounds (as in Lemma 3.3), we analyze the cumulative statistics of the estimator, enabling tighter variance bounds and improved overall complexity.

Due to space limitations, interested readers are referred to our full version [16] for the full analysis and framework details. Combining these refinements yields the bound stated in Theorem 1.4. □

4 Lower Bound of SSPPR-R Query

In this section, we prove the lower bound for SSPPR-R as stated in Theorem 1.6. As stated from **RoundWalks**, we have established an upper bound query complexity as $O(\log(1/\delta)/\delta)$. Since we can use $O(m)$ queries to acquire the entire graph, the total query complexity thus will not surpass $\Omega(\min(m, \log(1/\delta)/\delta))$, which is exactly the *optimal in arc-centric graph access model* of our result. The overall proof proceeds in three main steps: (i). Construct a family of hard-instance graphs with distribution; (ii). Reduce the SSPPR-R query to a simpler query type, i.e. examining certain amount of edges over the distribution; (iii). Derive the query complexity of the simplified problem over the distribution, based on observed known graph structures through graph oracle interactions.

Hard-instance Graph family. To construct a family of hard-instance graphs with accurate analysis of query complexity, we model different queries under uniform measurement. To achieve this, we keep the same node subsets and their in/out-degrees for all the graphs. In this setting, the query amount required to execute equals the known edge amount. Additionally, we need to reduce the SSPPR-R problem on this graph to a node-classification problem. To achieve this, we will construct a target node set X and a ground truth split $X = X_1 \sqcup X_2$, where nodes in X_1 and X_2 have marginal PPR scores violating the SSPPR-R requirement. Thus, any algorithm must predict this split. Formally, we define the graph family $\mathcal{U}(n, D, d)$ such that, for every sufficiently large n, it serves as a collection of hard instances for estimating SSPPR queries. This family is constructed based on a simple yet insightful core structure **U**, illustrated in Figure 1.

The graph is constructed from five disjoint node sets: $\{s\}$, Y_1, Y_2, X_1, and X_2. For convenience, let $Y = Y_1 \cup Y_2$ and $X = X_1 \cup X_2$. The node s is the designated *source* node. All nodes in X are treated as *target* nodes, whose PPR values with respect to s we aim to estimate. Each of the sets X_1, X_2, Y_1, and Y_2 contains $\Theta(n)$ nodes. We next describe how edges are constructed. Every node $y_1 \in Y_1$ has an incoming edge from the source node s. For each node $x_1 \in X_1$, we add D incoming edges from nodes in Y_1 and $D - d$ incoming edges from nodes in Y_2. Similarly, each node $x_2 \in X_2$ receives D incoming edges from Y_2 and $D - d$ incoming edges from Y_1. In addition, every node in X has a self-loop. With suitable choices of the parameters, the nodes in X_1 and X_2 will have noticeably different PPR values with respect to s. This separation will later be formalized in Lemma 4.1. To allow for more flexible control over the graph's parameters like node/edge count, we

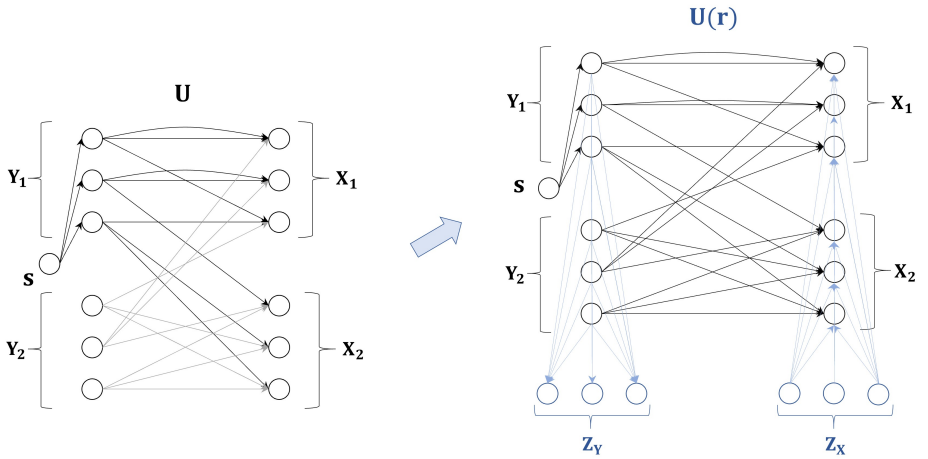


Fig. 1. Instance U $\longrightarrow$ r-padded Instance U(r).

next introduce a modified structure, the r-padded instance. Intuitively, information of the graphs is encoded by edges between Y and X. We can add some padding edges from nodes in Y (or to nodes in X) to confuse the algorithm. $\forall \mathrm{U} \in \mathcal{U}(n, D, d)$, we define the *$r$-padded* instance of U, denoted as $\mathrm{U}(r) = (V_{U(r)}, E_{U(r)})$ in the below by involving two independent node sets Z_X and Z_Y: (1) $V_{U(r)} = V_U \cup Z_X \cup Z_Y$, where Z_X, Z_Y are two additional node sets containing exactly r nodes; (2) $E_{U(r)} = E_U \cup \{(z_x, x) \mid \forall x \in X, z_x \in Z_X\} \cup \{(y, z_y) \mid \forall y \in Y, z_y \in Z_Y\} \cup \{(z_y, z_y) \mid \forall z_y \in Z_Y\}$, which incorporates extra edges coming from node set Z_x to X and Y to Z_Y, along with self-loops in Z_Y. Intuitively, when r is sufficiently large relative to D, each ADJ query originating from a node in X or Y may be misdirected to the auxiliary node sets Z_X or Z_Y with non-negligible probability. This allows us to increase the hardness for dense graphs without breaking core structure of $\mathcal{U}(n, D, d)$. Then we establish the existence of such hard-instance graph family below.

Lemma 4.1 [Existence of Hard-Instance Graph Family]. *Choose any constant $c \in (0, \frac{1}{2}]$ and any functions $\delta_0(n_0) \in (0,1)$, $m_0(n_0) \in \Omega(n_0) \cap O(n^2)$. For sufficiently large n_0, there exist parameters n, D, d, r (as function of n_0), such that for $\forall \mathrm{U}$ in $\mathcal{U}(n, D, d)$, its padded $\mathrm{U}(r)$ holds:*

(i). The node, edge count of $\mathrm{U}(r)$ is $O(n_0)$ and $O(m_0)$;

(ii). The edge count is in $\Theta(n(r+D)) = \Theta\left(\min(m_0, \log(\frac{1}{\delta_0})/\delta_0)\right)$, and the node count is in $\Theta(r+n) = O(n)$. Additionally, $D \geq \frac{3}{2}d$ and $d \leq \log(n)$;

(iii). For any $x_1, x_2 \in X_1, X_2$, the PPR scores $\pi(s, x_1), \pi(s, x_2)$ are fixed values, irrelevant to the choice of nodes or the graph instance. It holds that $\pi(s, x_1), \pi(s, x_2) \geq \delta_0, \pi(s, x_1) > \frac{1}{(1-c)^2} \cdot \pi(s, x_2)$.

This lemma forms the basis for our proof of the lower bound by contradiction. For any SSPPR-R algorithm with $o\left(\min(m_0, \log(1/\delta_0)/\delta_0)\right)$ queries, we will prove later that on our distribution of graphs over this family $\mathcal{U}(n, D, d)$ with r−padding, the algorithm must fail with high probability.

Distribution. We now formalize the generation of $\mathrm{U} \in \mathcal{U}(n, D, d)$ with node and edge labels, forming the distribution $\Sigma(n, D, d)$ over all label permutations (see Figure 2). Intuitively, $\Sigma(n, D, d)$ is created by randomly splitting the node set X into X_1 and X_2, then independently sampling all edge permutations consistent with the hard-instance template. The whole procedure is as follows:

(i). *Splitting X.* We first split the indices $\{1, \ldots, 2n\}$ into two equal parts, which determines whether each node $x_1, \ldots, x_{2n}$ belongs to X_1 or X_2. Each such split corresponds to choosing n indices for X_1, and all possible choices form the set $\binom{X}{n}$. For example, in Figure 2, the split SP $= \{1, 3\}$ places nodes x_1 and x_3 in X_1, while x_2 and x_4 belong to X_2.

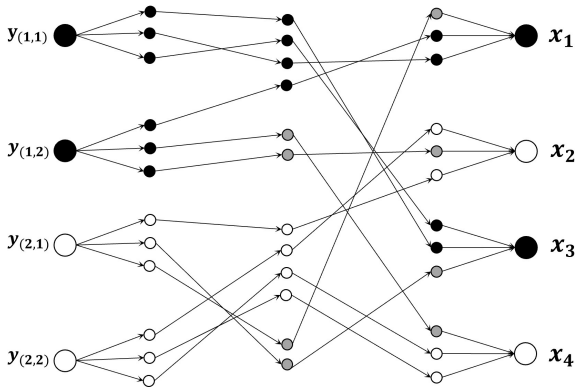


Fig. 2. An example for generating graph distribution $\Sigma(n, D, d) = \Sigma(2, 2, 1)$.

(ii). *Permuting in-edges of* X. Next, for each node $x_i \in X$, we independently permute its $2D - d$ incoming edges using a permutation $\mathrm{P}_i^{2D-d} \in \mathrm{S}^{2D-d}$. The permuted edges are then divided into two groups, corresponding to edges coming from Y_1 and Y_2. If $x_i \in X_1$, the first D edges in the permutation are assigned to Y_1 and the remaining $D - d$ edges to Y_2; the assignment is reversed when $x_i \in X_2$. We denote the collection of such permutations by $\mathrm{S}^{2D-d}(X)$. In Figure 2, applying $\mathrm{P}_3 = (1, 3, 2)$ assigns the first and third in-edges of x_2 to Y_2 (white-filled nodes) and the second to Y_1 (grey-filled nodes).

(iii). *Permuting out-edges of* Y_1, Y_2. We then permute the out-edges of Y_1 and Y_2 using two independent permutations $\mathrm{P}_{Y_1}^{n(2D-d)}$ and $\mathrm{P}_{Y_2}^{n(2D-d)}$. For a node $y_i^{(1)} \in Y_1$, its k-th outgoing edge connects to a node in X_1 if $\mathrm{P}_{Y_1}^{n(2D-d)}[(2D-d)(i-1)+k] \le nD$, and to a node in X_2 otherwise; the same rule applies symmetrically for nodes in Y_2. Figure 2 illustrates this step using the permutation $\mathrm{P}_{Y_1}^{n(2D-d)} = (1, 3, 2, 4, 5, 6)$, where the first four virtual nodes in the second column correspond to edges connecting to nodes in X_1.

(iv). *Connecting* X *and* Y. Finally, we match the out-edges of nodes in Y with the in-edges of nodes in X. For example, for the pair Y_1 and X_1, we select a random bijection $\mathrm{S}(Y_1, X_1) : \{1, \ldots, nD\} \to \{1, \ldots, nD\}$ that maps the out-edges of Y_1 to the in-edges of X_1. Similar bijections (such as $\mathrm{S}(Y_2, X_1)$) are used for the other pairs. Each edge is represented as a tuple $(y_{i_j}^{(1)}, k_j)$ or $(x_{i'_j}^{(1)}, k'_j)$, and the bijection specifies how these tuples are matched. In Figure 2, the bijection $\mathrm{S}(Y_1, X_1) = (4, 3, 2, 1)$ connects the corresponding virtual nodes (black-filled) across the columns.

Following the above procedure, we obtain a graph instance $\mathrm{U} \in \mathcal{U}(n, D, d)$.

Lemma 4.2. *For all possible* $\mathrm{SP}, \mathrm{P}_i^{2D-d}, \mathrm{P}_{Y_1}^{n(2D-d)}, \mathrm{P}_{Y_2}^{n(2D-d)}, \mathrm{P} \in \binom{X}{n} \times \mathrm{S}^{2D-d}(X) \times \mathrm{S}^{n(2D-d)}(Y_1) \times \mathrm{S}^{n(2D-d)}(Y_2) \times \Pi_{Y_a \in \{Y_1, Y_2\}, X_b \in \{X_1, X_2\}} S(Y_a, X_b)$*: the above procedure links to a legal graph instance* $\mathrm{U} \in \mathcal{U}(n, D, d)$.

Based on above Lemma, we can then construct our distribution over splits and bijections as:

Definition 4.3. $\Sigma(n, D, d)$ is defined as the uniform distribution over $\binom{X}{n} \times \mathrm{S}^{2D-d}(X) \times \mathrm{S}^{n(2D-d)}(Y_1) \times \mathrm{S}^{n(2D-d)}(Y_2) \times \Pi_{Y_i \in \{Y_1, Y_2\}, X_i \in \{X_1, X_2\}} S(Y_i, X_i)$. For $\sigma \sim \Sigma(n, D, d)$, we use U_σ to denote the corresponding labeled graph instance $\mathrm{U} \in \mathcal{U}(n, D, d)$ and $\mathrm{U} \sim \Sigma(n, D, d)$ to denote a sample of this distribution.

Establishing the Main Theorem. Now that the distribution $\Sigma(n, D, d)$ is formally defined, we can prove the target lower bound by contradiction: no SSPPR-R algorithm can have both a bounded query complexity and a constant success probability on $\Sigma(n, D, d)$.

We begin by addressing two technical issues. First, standard SSPPR-R algorithms are formulated for simple graphs, while instances drawn from $\Sigma(n, D, d)$ often contain parallel edges. Second,

Lemma 4.1 is stated for padded graphs, whereas our hard distribution is defined on the original, unpadded instances. To handle these issues, we use a two-step reduction. We first lift an arbitrary algorithm for simple graphs so that it operates on the padded multigraph instances $\mathrm{U}(r)$. We then restrict this lifted algorithm back to the unpadded graph, thereby obtaining the desired adapted algorithm on $\Sigma(n, D, d)$. Moreover, by construction of $\mathcal{U}(n, D, d)$, the nodes in X_1 and X_2 exhibit a significant gap in their PPR scores. Consequently, any algorithm that satisfies the relative-error guarantee for SSPPR-R must be able to recover the ground truth partition of X into $X_1 \sqcup X_2$. We formalize both ingredients in the following lemma.

Lemma 4.4. *Assume that there is an SSPPR-R algorithm $\mathcal{A}$ on simple directed graphs with expected query complexity $T = o(\min(m_0, \log(1/\delta_0)/\delta_0))$ with threshold δ, error parameter c and failure probability bound p_f. Then for any constant γ, there are sufficiently large parameters n, D, d along with an algorithm $\mathcal{A}_{ad}$. Using at most γnD arc-centric queries on each $\mathrm{U} \sim \Sigma(n, D, d)$, $\mathcal{A}_{ad}$ will output the ground truth split of $X = X_1 \sqcup X_2$ with average success probability at least $1 - 2p_f$.*

With this lemma in hand, we show that the probability that $\mathcal{A}_{ad}$ recovers the correct partition is sufficiently small to obtain a contradiction.

To analyze the query complexity limit, we must first formally measure the information any algorithm receives. Each graph in the family $\mathcal{U}(n, D, d)$ is uniquely determined by the connections between Y and X. Each such connecting edge has two labels: the "k^{th} out-edge of y_i" and the "k'^{th} in-edge of x_j". This can be viewed as connecting two half-edges. An arc-centric query reveals partial information of one such connection. Formally, a known edge $e_t = [(y_i, k), (x_j, k')]$ is an event in $\Sigma(n, D, d)$ representing that the k^{th} out-edge of y_i connects to the k'^{th} in-edge of x_j. In its execution, our algorithm $\mathcal{A}_{ad}$ sequentially observes the events, in other words, knows the edges $e_1, \ldots, e_T$, and its task is to infer the most probable split based on this information. The optimal strategy for any algorithm is to output the split that maximizes the posterior probability (i.e., the MAP estimator):

$$\mathrm{argmax}_{\mathrm{SP}}\, \mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[\mathrm{SP \text{ is the correct split of } U} \mid e_1, \ldots, e_T].$$

Therefore, the expected success probability of any algorithm using at most T queries is bounded as:

Lemma 4.5 [Success Probability Expression]. *For any algorithm $\mathcal{A}_{ad}$ using at most $T = \gamma nD$ arc-centric queries on an instance $\mathrm{U} \sim \Sigma(n, D, d)$, its expected probability of outputting the correct split of X is bounded by*

$$\max_{e_1,\cdots,e_T} \max_{\mathrm{SP}} \mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[\mathrm{SP} \textit{ is the correct split of } \mathrm{U} \mid e_1, \cdots, e_T].$$

To estimate $\mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[\mathrm{SP \text{ is the correct split of } U} \mid e_1, \cdots, e_T]$, we rely on the following two observations: (i). **Traversal is Hard**: An algorithm that tends to cover the entire graph needs to know $\Omega(nD)$ edges. (ii). **Random Walk is Hard**: Consider an algorithm that classifies each node x_i into X_1 or X_2 in the ground truth split, by sampling random incident edges. Observe that nodes in X_1 differ from those in X_2 because they have d more edges originating from Y_1 among their $2D - d$ in-edges. Then, to achieve a failure probability of p_i, the algorithm must sample $\Omega(\frac{D}{d} \log_{D/(D-d)}(1/p_i)) \subset \Omega(D)$ edges. Specifically, to guarantee a uniform error bound $p_i \in O(1/n)$, , the required number of samples becomes $\Omega(D)$ per node.

With these observations, we then prove the following lemma:

Lemma 4.6 [Bounded Success Probability]. *There is a constant γ such that for sufficiently large n, $d \le \log(n)$, $D > \frac{3d}{2}$, and $T \le \gamma nD$, the following holds. Given any known edges $e_1, \ldots, e_T$ and any split SP of X, we have: $\mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[\mathrm{SP} \textit{ is the correct split of } \mathrm{U} \mid e_1, \cdots, e_T] \le \frac{1}{n}$.*

Proof Sketch. With $o(nD)$ known edges, for most nodes $x_i \in X$, we only know a small fraction of their in-edges, as per the "Traversal is Hard" intuition. For the split SP, there exists a subset

$X' = X'_1 \cup X'_2$, where $X'_1 \subset X_1$, $X'_2 \subset X_2$, and $|X'_1|, |X'_2| \in \Omega(n)$, such that each $x_i \in X'$ appears at most $o(D)$ times in $e_1, \cdots, e_T$. We then use the 'Random Walk is Hard' intuition to show that, for nodes in X', it is hard for the algorithm to determine their class in the split. For each pair $x_{i1} \in X'_1, x_{i2} \in X'_2$, consider a "switched" split SP′ created by setting $x_{i1} \in X_2$ and $x_{i2} \in X_1$, while fixing the split for all other nodes. Using a Bayesian approach, we can bound the target success probability $\mathbb{P}[\dots]$ by comparing it to the probabilities of such switched splits. As split SP is sampled from uniform distribution, applying Bayes' rule gives

$$\mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[\text{SP is the correct split of U} \mid e_1, \cdots, e_T] = \frac{\mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[e_1, \cdots, e_T \mid \mathrm{SP}]}{\sum_{\mathrm{SP}'} \mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[e_1, \cdots, e_T \mid \mathrm{SP}']}.$$

Notice that there are $\Omega(n^2)$ such switched splits. The proof reduces to showing that for any such switched split SP′: $\frac{\mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[e_1,\cdots,e_T|\mathrm{SP}]}{\mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[e_1,\cdots,e_T|\mathrm{SP}']} \in O(n^{1-\epsilon})$. We can then decompose this ratio into a product of terms, one for each edge e_t: $\prod_{t=1}^{T} \frac{\mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[e_t|\mathrm{SP},e_1,\dots,e_{t-1}]}{\mathbb{P}_{\mathrm{U}\sim\Sigma(n,D,d)}[e_t|\mathrm{SP}',e_1,\dots,e_{t-1}]}$. Intuitively, when the known edge e_t is not connected to the switched nodes x_{i1} or x_{i2}, it is unrelated to the switch, and the ratio is 1. For the remaining $2\gamma D$ known edges, we can establish that each such ratio is bounded by $1 + O(d/D)$. Finally, since $d \in O(\log(n))$, the total ratio is bounded by $(1 + O(d/D))^{2\gamma D} \in O(n^{1-\epsilon})$, which leads to the result. By Lemma 4.6, we conclude that the expected success probability of $\mathcal{A}_{ad}$ is less than $\frac{1}{n}$. This contradicts the guarantee from Theorem 4.4 (which stated a success probability of at least $1 - 2p$). This contradiction shows that the initial assumption—the existence of an algorithm $\mathcal{A}$ with $T \in o\left(\min(m_0, \log(1/\delta_0)/\delta_0)\right)$ queries—must be false. Therefore, any such algorithm must require $\Omega\left(\min(m_0, \log(1/\delta_0)/\delta_0)\right)$ queries in expectation. This completes the proof of our Theorem 1.6. □

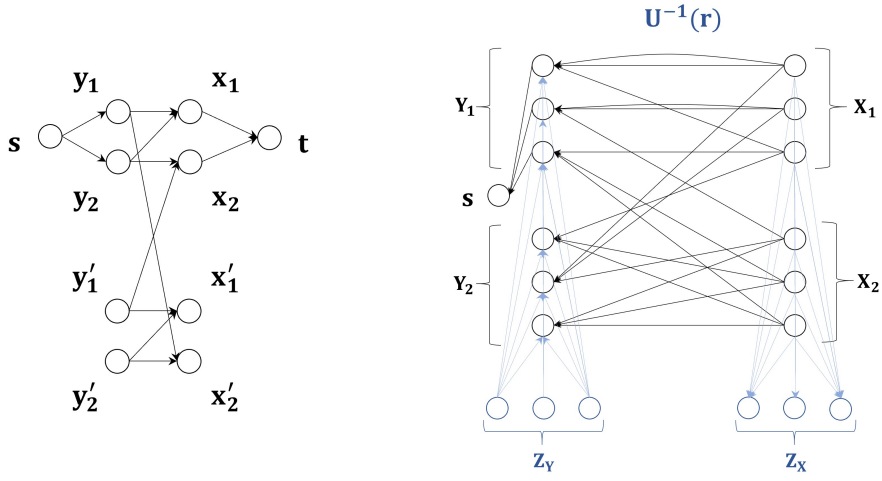
5 Lower Bound of SSPPR-A Query

In this section, we present the proof of Theorem 1.8. In contrast to the SSPPR-R case, the hardness in the absolute-error setting stems not from fundamental sampling variance when estimating a large PPR value $\pi(s,t)$. An estimator of $\pi(s,t)$ based on N random walks behaves approximately as $\mathcal{N}(\pi(s,t), \sigma^2)$ with standard deviation $\sigma = \Theta(\pi(s,t)N^{-1/2})$. Thus, when $\pi(s,t) = \Theta(1)$, achieving an absolute error of ε with constant success probability inherently requires $N = \Omega(1/\varepsilon^2)$ walks—an unavoidable statistical barrier. Our proof embeds a hidden binary matrix into the graph and links $\pi(s,t)$ to the count of 1 in the matrix. Since each bit is Bernoulli, the total count of ones follows a binomial law and is well approximated by a normal distribution, paralleling the distribution of the random-walk estimator. Then, any algorithm must query a large fraction of the graph to recover this count; otherwise, its estimate of $\pi(s,t)$ incurs statistical uncertainty exceeding ε.

Graph Family and Distribution. We define a graph family $\mathcal{G}(D, \mathbf{b})$, where D is a positive integer and $\mathbf{b}$ is a $D \times D$ binary matrix with entries b_{ij}. Each graph contains $4D + 2$ vertices, partitioned into six disjoint sets $\{s\} \cup \{t\} \cup X \cup Y \cup X' \cup Y'$, where s and t are the designated *source* and *target* nodes, respectively, and each of the sets X, Y, X', Y' contains exactly D nodes. Edges are added as follows. The source node s connects to every node in Y, and every node in X connects to the target node t. For every pair of indices (i, j), the connections between the sets (Y, Y') and (X, X') are determined by the matrix entry b_{ij}:

- if $b_{ij} = 1$, we add edges (y_i, x_j) and (y'_i, x'_j);
- if $b_{ij} = 0$, we add edges (y_i, x'_j) and (y'_i, x_j).

Thus, the matrix $\mathbf{b}$ controls how nodes in Y and Y' are wired to nodes in X and X'. This $\mathbf{b}$-dependent wiring introduces structured randomness that is crucial for constructing hard instances in our lower-bound proofs. An example is illustrated in Figure 3(a), which shows $\mathcal{G}(D, \mathbf{b})$ with $D = 2$ and $\mathbf{b} := \mathbf{b}(1, \cdot) = (1, 0)$ and $\mathbf{b}(2, \cdot) = (1, 1)$.



(a) Distribution $\mathcal{G}(D, \mathbf{b})$ (b) Instance $\mathcal{U}^{-1}(r)$

Fig. 3. Graph examples for lower bounds. (a) denotes the example of generating graph distribution $\mathcal{G}(D, \mathbf{b})$ in SSPPR-A query and (b) represents the graph instance $\mathcal{U}^{-1}(r)$ derived from $\mathcal{U}(r)$ in STPPR query.

Lemma 5.1 [Properties of $\mathcal{G}(D, \mathbf{b})$]. *For any instance $\mathcal{G}(D, \mathbf{b})$, it holds:*

(i). The graph contains $4D + 2$ nodes and $2D(D + 1)$ edges.

(ii). The out-degree of each node in Y or Y' is D. The in-degree of each node in X or X' is D.

(iii). $\pi(s, t) = \alpha \left(\sum_{i,j=1}^{D} b_{ij} \right) \frac{(1-\alpha)^3}{D^2}$.

We now define $\Sigma(D)$, a probability distribution over the graph family $\mathcal{G}(D, \mathbf{b})$, which serves as our hard distribution for the lower bound. For each fixed D, every entry b_{ij} of the binary matrix $\mathbf{b}$ is chosen independently from a Bernoulli distribution Bern$(1/2)$. Under this distribution, the total number of ones $S := \sum_{i,j=1}^{D} b_{ij}$ follows a binomial law Binomial$(D^2, 1/2)$. For node and edge indexing, we adopt the natural scheme: the i-th outgoing edge of s connects to y_i, the j-th incoming edge of t originates from x_j, and for each y_i, its j-th outgoing edge goes to x_j or x'_j depending on the value of b_{ij}. Combining all the above preparations, we now proceed to prove the lower bound. As in the relative-error case, the proof naturally decomposes into two key lemmas: (i) establishing the existence of an appropriate hard distribution and its parameters, and (ii) analyzing the conditional distribution of outcomes under a bounded number of oracle queries.

Lemma 5.2. *For any constants $\alpha \in (0, 1)$, $p \in (0, 1)$, and $m_0 \in \Omega(n_0), m_0 \leq \binom{n_0}{2}, \varepsilon_0 \in (0, 1))$ as function of n_0 and assuming ε_0 is sufficiently small when n_0 is sufficiently large, then there exists constants $\gamma > 0$ such that for any sufficiently small absolute error tolerance $\varepsilon > 0$, we can choose an integer $D(n_0)$ satisfying:*

(i). The graph $\mathcal{G}(D, \mathbf{b})$ has no more than n_0 nodes and no more then m_0 edges.

(ii). If two matrices $\mathbf{b}$ and $\mathbf{b}'$ have sums $S = \sum b_{ij}$ and $S' = \sum b'_{ij}$ such that $|S - S'| \geq \gamma D$, then the corresponding PPR values satisfy $|\pi(s, t) - \pi'(s, t)| > 2\varepsilon$.

(iii). $D^2 = \Omega(\min(1/\varepsilon_0^2, m_0))$.

Then we develop the next lemma to formalizes the statistical difficulty of estimating the sum S with the precision required for our overall lower bound.

Lemma 5.3. *For any constant $p \in (0, 1)$, there exists a constant $\gamma' > 0$ such that for a sufficiently large D, any algorithm making at most $T \leq D^2/2$ queries to a graph drawn from $\Sigma(D)$ will fail to*

estimate $S = \sum_{i,j=1}^{D^2} b_{ij}$ *with an error less than* $\gamma' D$. *Specifically, if* $\hat{S}$ *is the algorithm's estimate, then it holds:* $\mathbb{P}_{\mathbf{b}\sim\Sigma(D)}[|S - \hat{S}| > \gamma' D] > p$.

Combining the constructed D in Lemma 5.2, we conclude that using only $o(\min(1/\varepsilon_0^2, m_0))$ queries will cause larger failure probability than p in evaluating $\pi(s,t)$ with absolute error guarantee. □

6 Lower Bound of STPPR Query

In this section, we present the proof of Theorem 1.7. The overall argument follows the same structure as the proof of Theorem 1.6. The key difference lies in the construction of the hard-instance graph family: to create the required discrepancy in single target PPR values, we reverse all edge directions in the base graph U, obtaining its inverse family U^{-1}. The details of this construction and its properties are formalized in Lemma 6.1:

Lemma 6.1 [STPPR]. *Let* U^{-1} *denote the inverse of a graph* U *obtained by reversing the direction of each edge. Choose any decay factor* $\alpha \in (0,1)$, *error parameter* $c \le \frac{1}{2}$, *and functions* $\delta_0(n_0) \in \left(0, \frac{(1-\alpha)^2}{16}\right)$ *and* $m_0(n_0) \in \Omega(n_0) \cap O(n_0^2)$. *For sufficiently large* n_0, *there exists graph parameters* n, D, d *(as functions of* n_0*) such that for* $\forall \mathrm{U} \in \mathcal{U}(n, D, d)$, *the following properties hold for the inverse* U^{-1} *that:*

(i). The node, edge count of U^{-1} *is* $O(n_0), O(m_0)$.

(ii). $n^{1/2} \ge D \ge 2d, d \le \log n$ *and* $Dn = \Omega\left(\min\left(m_0, n_0 \log n_0 \cdot \frac{1}{\delta_0}\right)\right)$.

(iii). For all $x_1, x_2 \in X_1, X_2$ *satisfying* $\pi(x_1, s), \pi(x_2, s) \ge \delta_0$, *we have* $\pi(x_1, s) \ge \frac{1}{(1-c)^2}\pi(x_2, s)$.

As in the proof of Theorem 1.6, estimating $\pi(x_i, s)$ necessarily requires identifying the correct partition of X into X_1 and X_2. In the arc-centric query model, a directed graph and its edge-reversed version are indistinguishable; hence we may directly apply Lemma 4.6 to the inverse family $\mathcal{U}^{-1}(n, D, d)$. It follows that any algorithm for STPPR must perform enough queries to recover the hidden partition, implying a lower bound of $\Omega\left(\min\left(m_0,\ n_0 \log n_0 \cdot \frac{1}{\delta_0}\right)\right)$, establishing Theorem 1.7. □

7 Conclusion

In this work, we significantly narrow or even close the gaps between the upper and lower bounds for the computational complexity of SSPPR queries. On the upper bound side, we tighten the complexities of SSPPR-A and SSPPR-R to $O(1/\epsilon^2)$ and $O\left(\min\left(\log(1/\delta)/\delta,\ m + n\log n\ \log(\frac{\log n}{m\delta})\right)\right)$, respectively. These results improve the best known bounds by factors of $\log(1/\epsilon)$ and $\log(\frac{\log n}{m\delta})$. On the lower bound side, under the standard arc-centric model, we prove stronger bounds of $\Omega(\min(m, 1/\epsilon^2))$ for SSPPR-A and $\Omega(\min(m, \log(1/\delta)/\delta))$ for SSPPR-R. These improve previous results by factors of m/n or $1/\epsilon$, and m/n or $\log(1/\delta)$, respectively. Together, these results substantially strengthen the theoretical foundations of PPR computation. For SSPPR-R, the upper and lower bounds coincide for graphs with $m \in \Omega(n\log^2 n)$ and thresholds satisfying $1/\delta \in O(\mathrm{poly}(n))$, establishing theoretical optimality under most graph regimes. For SSPPR-A, partial optimality is obtained for coarse absolute error requirements (that is, larger ϵ), where the upper bound reduces to $O(1/\epsilon^2)$, matching the lower bound. To the best of our knowledge, this is the first work establishing optimal algorithmic results for SSPPR queries, establishing a new theoretical foundation for the study of PPR computation.

Acknowledgments

This research / project is supported by the National Research Foundation, Singapore, under its AI Singapore Programme (AISG Award No: AISG3-RP-2024-034), and under its Frontier CRP Grant (NRF-F-CRP-2024-0005). This research is also supported by NTU SUG-NAP, and supported by the Ministry of Education, Singapore, under an AcRF Tier 2 Grant (MOE-000761-01). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the National Research Foundation, Singapore.

References

[1] Konstantin Avrachenkov, Nelly Litvak, Danil Nemirovsky, and Natalia Osipova. 2007. Monte Carlo methods in PageRank computation: When one iteration is sufficient. *SIAM J. Numer. Anal.* 45, 2 (2007), 890–904.

[2] Christian Bertram, Mads Vestergaard Jensen, Mikkel Thorup, Hanzhi Wang, and Shuyi Yan. 2025. Estimating Random-Walk Probabilities in Directed Graphs. *arXiv preprint arXiv:2504.16481* (2025).

[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] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30, 1-7 (1998), 107–117.

[5] Douglas A Drossman, Lin Chang, Jill K Deutsch, Alexander C Ford, Albena Halpert, Kurt Kroenke, Samuel Nurko, Johannah Ruddy, Julie Snyder, and Ami Sperber. 2021. A review of the evidence and recommendations on communication skills and the patient–provider relationship: a Rome foundation working team report. *Gastroenterology* 161, 5 (2021), 1670–1688.

[6] Dániel Fogaras, Balázs Rácz, Károly Csalogány, and Tamás Sarlós. 2005. Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. *Internet Mathematics* 2, 3 (2005), 333–358.

[7] Oded Goldreich. 1999. Combinatorial property testing (a survey). *Randomization Methods in Algorithm Design* 43 (1999), 45–59.

[8] Oded Goldreich. 2008. Computational complexity: a conceptual perspective. *ACM Sigact News* 39, 3 (2008), 35–39.

[9] Oded Goldreich. 2017. *Introduction to property testing*. Cambridge University Press.

[10] Oded Goldreich, Shari Goldwasser, and Dana Ron. 1998. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)* 45, 4 (1998), 653–750.

[11] Oded Goldreich and Dana Ron. 1997. Property testing in bounded degree graphs. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (STOC)*. 406–415.

[12] Taher H Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*. 517–526.

[13] Guanhao Hou, Xingguang Chen, Sibo Wang, and Zhewei Wei. 2021. Massively parallel algorithms for personalized pagerank. *Proceedings of the VLDB Endowment (PVLDB)* 14, 9 (2021), 1668–1680.

[14] Guanhao Hou, Qintian Guo, Fangyuan Zhang, Sibo Wang, and Zhewei Wei. 2023. Personalized PageRank on Evolving Graphs with an Incremental Index-Update Scheme. *Proceedings of the ACM on Management of Data (SIGMOD)* 1, 1 (2023), 1–26.

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

[16] Xinpeng Jiang, Haoyu Liu, Siqiang Luo, and Xiaokui Xiao. 2025. Near-Optimality for Single-Source Personalized PageRank. *arXiv preprint arXiv:2507.14462* (2025).

[17] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. *International Conference on Learning Representations* (2018).

[18] Meihao Liao, Rong-Hua Li, Qiangqiang Dai, Hongyang Chen, Hongchao Qin, and Guoren Wang. 2023. Efficient Personalized PageRank Computation: The Power of Variance-Reduced Monte Carlo Approaches. *Proceedings of the ACM on Management of Data (SIGMOD)* 1, 2 (2023), 1–26.

[19] Meihao Liao, Rong-Hua Li, Qiangqiang Dai, Hongyang Chen, Hongchao Qin, and Guoren Wang. 2023. Efficient personalized pagerank computation: The power of variance-reduced monte carlo approaches. *Proceedings of the ACM on Management of Data (SIGMOD)* 1, 2 (2023), 1–26.

[20] Meihao Liao, Rong-Hua Li, Qiangqiang Dai, and Guoren Wang. 2022. Efficient personalized PageRank computation: A spanning forests sampling based approach. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD)*. 2048–2061.

[21] Meihao Liao, Rong-Hua Li, Qiangqiang Dai, and Guoren Wang. 2022. Efficient personalized pagerank computation: A spanning forests sampling based approach. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD)*. 2048–2061.

[22] Ningyi Liao, Dingheng Mo, Siqiang Luo, Xiang Li, and Pengcheng Yin. 2022. SCARA: Scalable Graph Neural Networks with Feature-Oriented Optimization. In *Proceedings of the VLDB Endowment* (Sydney, Australia), Vol. 15. VLDB Endowment, 3240–3248. https://doi.org/10.14778/3551793.3551866

[23] Ningyi Liao, Dingheng Mo, Siqiang Luo, Xiang Li, and Pengcheng Yin. 2024. Scalable decoupling graph neural network with feature-oriented optimization. *The VLDB Journal* 33, 3 (2024), 667–683.

[24] Haoyu Liu and Siqiang Luo. 2024. BIRD: Efficient Approximation of Bidirectional Hidden Personalized PageRank. *Proceedings of the VLDB Endowment (PVLDB)* 17, 9 (2024), 2255–2268.

[25] Qin Liu, Zhenguo Li, John CS Lui, and Jiefeng Cheng. 2016. Powerwalk: Scalable personalized pagerank via random walks with vertex-centric decomposition. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 195–204.

[26] Peter Lofgren, Siddhartha Banerjee, and Ashish Goel. 2015. Bidirectional pagerank estimation: From average-case to worst-case. In *Algorithms and Models for the Web Graph: 12th International Workshop, WAW 2015, Eindhoven, The Netherlands, December 10-11, 2015, Proceedings 12*. Springer, 164–176.

[27] Peter A Lofgren, Siddhartha Banerjee, Ashish Goel, and Comandur Seshadhri. 2014. Fast-ppr: Scaling personalized pagerank estimation for large graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1436–1445.

[28] Shangqi Lu, Wim Martens, Matthias Niewerth, and Yufei Tao. 2022. Optimal algorithms for multiway search on partial orders. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 175–187.

[29] Siqiang Luo, Xiaokui Xiao, Wenqing Lin, and Ben Kao. 2019. BATON: Batch one-hop personalized PageRanks with efficiency and accuracy. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 32, 10 (2019), 1897–1908.

[30] Wim Martens and Tina Popp. 2022. The complexity of regular trail and simple path queries on undirected graphs. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 165–174.

[31] Phuong Nguyen, Paolo Tomeo, Tommaso Di Noia, and Eugenio Di Sciascio. 2015. An evaluation of simrank and personalized pagerank to build a recommender system for the web of data. In *Proceedings of the 24th international conference on world wide web*. 1477–1482.

[32] Amund Tveit. 2003. On the complexity of matrix inversion. *Mathematical Note* 1 (2003).

[33] Andrea Vattani, Deepayan Chakrabarti, and Maxim Gurevich. 2011. Preserving personalized pagerank in subgraphs. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer, 793–800.

[34] Hanzhi Wang, Mingguo He, Zhewei Wei, Sibo Wang, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2021. Approximate graph propagation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1686–1696.

[35] Hanzhi Wang and Zhewei Wei. 2023. Estimating Single-Node PageRank in $\tilde{O}(min\{d_t, \sqrt{m}\})$ Time. *Proceedings of the VLDB Endowment (PVLDB)* 16, 11 (2023), 2949–2961.

[36] 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.

[37] Hanzhi Wang, Zhewei Wei, Ji-Rong Wen, and Mingji Yang. 2024. Revisiting Local Computation of PageRank: Simple and Optimal. In *Proceedings of the 56th Annual ACM symposium on Theory of computing (STOC)*. 911–922.

[38] Runhui Wang, Sibo Wang, and Xiaofang Zhou. 2019. Parallelizing approximate single-source personalized PageRank queries on shared memory. *The VLDB Journal* 28, 6 (2019), 923–940.

[39] Sibo Wang, Renchi Yang, Runhui Wang, Xiaokui Xiao, Zhewei Wei, Wenqing Lin, Yin Yang, and Nan Tang. 2019. Efficient algorithms for approximate single-source personalized pagerank queries. *ACM Transactions on Database Systems (TODS)* 44, 4 (2019), 1–37.

[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] Zhewei Wei, Xiaodong He, Xiaokui Xiao, Sibo Wang, Shuo Shang, and Ji-Rong Wen. 2018. Topppr: top-k personalized pagerank queries with precision guarantees on large graphs. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD)*. 441–456.

[42] Zhewei Wei, Ji-Rong Wen, and Mingji Yang. 2024. Approximating single-source Personalized PageRank with absolute error guarantees. In *Proceedings of the 27th International Conference on Database Theory*, Vol. 290. 9:1–9:19. https://doi.org/10.4230/LIPIcs.ICDT.2024.9

[43] Hao Wu, Junhao Gan, Zhewei Wei, and Rui Zhang. 2021. Unifying the global and local approaches: an efficient power iteration with forward push. In *Proceedings of the 2021 International Conference on Management of Data*. 1996–2008.

[44] Hao Wu, Junhao Gan, Zhewei Wei, and Rui Zhang. 2021. Unifying the global and local approaches: an efficient power iteration with forward push. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD)*. 1996–2008.

[45] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, Philip S Yu, et al. 2008. Top 10 algorithms in data mining. *Knowledge and information systems* 14 (2008), 1–37.

[46] Renchi Yang. 2022. Efficient and Effective Similarity Search over Bipartite Graphs. In *Proceedings of the ACM Web Conference (WWW) 2022*. 308–318.

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

[48] Minji Yoon, Jinhong Jung, and U Kang. 2018. Tpa: Fast, scalable, and accurate method for approximate random walk with restart on billion scale graphs. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 1132–1143.

[49] Junjie Zhou, Meihao Liao, Rong-Hua Li, Longlong Lin, and Guoren Wang. 2025. One Index for All: Towards Efficient Personalized PageRank Computation for Every Damping Factor. *Proceedings of the ACM on Management of Data (SIGMOD)* 3, 4 (2025), 1–26.

[50] Zulun Zhu, Sibo Wang, Siqiang Luo, Dingheng Mo, Wenqing Lin, and Chunbo Li. 2024. Personalized PageRanks over Dynamic Graphs–The Case for Optimizing Quality of Service. In *Proceedings of the 2024 IEEE 40th International Conference on Data Engineering (ICDE)*.

Received December 2025; accepted February 2026