Differentially Oblivious Graph Ball Queries

Zimai Guo[1], Cheng Hong[2], Zhechen Li[2,⋆]

[1]School of Computer Science, Peking University, Beijing, China
[2]Ant Group, Beijing, China
{endergzm,lizhechen}@pku.edu.cn, vince.hc@antgroup.com

Abstract. A graph ball query returns all vertices within distance r of a source vertex. On N-vertex graphs with maximum degree at most Δ, ordinary breadth-first search answers such queries with $\mathrm{O}_{\Delta,r}(1)$ record accesses. These accesses depend on private edges and can reveal graph structure. Full obliviousness hides this dependence, but incurs substantial overhead. Differential obliviousness (DO) [Chan et al., SODA'2019] trades off between privacy and efficiency by relaxing full obliviousness to (ε, δ)-differential privacy on the access traces of neighboring graphs.
We study DO graph ball queries under three widely-adopted edge-level neighboring relations: add/delete, replacement, and 2-switch. Starting from padded adjacency records, we charge all observable accesses, including preprocessing. For fixed $\Delta, r \geq 2$, error probability $0 < \beta < 1/2$, and $0 \leq \delta \leq (1-2\beta)/2$, we prove an $\Omega_{\Delta,r,\beta}(1 + Ne^{-\varepsilon})$ expected lower bound at every sufficiently large connected graph and every public source. For every fixed $r \geq 2$, we give $(\varepsilon, 0)$-DO sampling algorithms with expected cost $\mathrm{O}_{\Delta,r,\beta}(1 + Ne^{-\varepsilon/C})$ using constant size of hidden workspace, where C depends only on Δ, r and the neighboring relation. Especially, for $r = 2$, we have $C = 1$, which indicates that the algorithm achieves the tight bound under all three relations. Finally, we consider repeated queries on the same graph, where the cost of protected preprocessing can be amortized across queries. For fixed Δ, r, β, we construct a linear-size reusable index with $(\varepsilon, 0)$-DO preprocessing and constant query cost. The expected preprocessing cost is linear in N when ε exceeds a sufficiently large constant multiple of $\log N$.

Keywords: Differential obliviousness · graph algorithms · access patterns

1 Introduction

Graph ball queries are basic operations in graph databases and social network analysis. One-hop and two-hop neighborhoods appear explicitly in the LDBC Social Network Benchmark [7], and neighborhood queries are among the most commonly reported graph computations in the survey of Sahu et al. [18]. Local neighborhoods also play a role in correlation-decay algorithms for approximate counting, which use local marginal estimates to count independent sets and

⋆ Authors are listed in lexicographical order. Zhechen Li is the corresponding author.

matchings [19,2]. Let $G = (V, E)$ be a N-vertex unweighted graph with maximum degree at most Δ. Given a center $s \in V$ and a radius r, a *graph ball* is

$$\mathsf{B}_r(G, s) := \{v \in V : \mathrm{dist}_G(s, v) \leq r\}.$$

Ordinary breadth-first search (BFS) computes $\mathsf{B}_r(G, s)$ by expanding the newly discovered frontier for r rounds. For fixed Δ and r, it reads only $\mathrm{O}_{\Delta,r}(1)$ adjacency records, independently of N. Its efficiency comes from visiting only vertices near s. However, the addresses of the records it reads depend on private edges, so the access pattern can reveal graph structure even if the records themselves are encrypted. The risks of such leakage are illustrated by access pattern attacks on searchable encryption [13].

Full obliviousness hides this dependence by requiring access traces to be statistically indistinguishable across inputs with the same public parameters. Generic oblivious RAM simulation [11] provides a way to hide memory accesses, but introduces logarithmic overhead. *Differential obliviousness (DO)* relaxes full obliviousness by requiring the access trace to satisfy (ε, δ)-differential privacy [6] on neighboring inputs. Chan et al. [4] developed foundations for this notion and showed that it can improve efficiency for tasks such as sorting, merging, and range queries. This raises the possibility of retaining the efficiency of local graph exploration under a relaxed privacy guarantee.

However, for graphs, there has already been evidence that DO can make explorations expensive. Beimel et al. [3] show that connectivity testing on bounded-degree graphs can require $\Omega(\sqrt{N})$ probes for constant privacy budget and sufficiently small δ, though ordinary testers use constant number of probes with respect to N. This raises the question of whether DO also forces non-local accesses when the task is to return an exact local neighborhood. For graph ball queries with fixed Δ, r, both the answer size and the cost of ordinary exploration are bounded independently of N. This motivates our central question.

Can differential obliviousness preserve the locality of graph ball queries?

1.1 Our contributions

We answer this question by establishing lower and upper bounds on the access complexity of DO graph ball queries in cell-probe model. Starting from padded adjacency records, we charge all input reads and observable workspace accesses in both preprocessing and query phases. We consider three edge-level neighboring relations: adding/deleting one edge, replacing one edge, and applying a 2-switch.

We establish universal lower bounds and give sampling algorithms that attain the optimal dependence on the privacy budget for two-hop queries. The following informal theorem summarizes these results.

Main Theorem (Informal). *Fix $\Delta, r \geq 2$ and $0 < \beta < 1/2$. For all sufficiently large N and any $\varepsilon \geq 0$ and $0 \leq \delta \leq (1 - 2\beta)/2$, the following hold under each of the three neighboring relations. All algorithms are required to return $\mathsf{B}_r(G, s)$ with probability at least $1 - \beta$ on every connected N-vertex graph G of maximum degree at most Δ and every public source s.*

1. ***Lower bound.*** *Every (ε, δ)-DO algorithm incurs $\Omega_{\Delta,r,\beta}(1+Ne^{-\varepsilon})$ expected input reads for every G, s.*
2. ***Upper bound.*** *There exists a pure ε-DO algorithm with $\mathrm{O}_{\Delta,r,\beta}(1+Ne^{-\varepsilon/C})$ expected total accesses for every G, s, where C depends only on Δ, r and the neighboring relation.*

In particular, the bounds are tight when $r = 2$.

Further, we also construct a reusable index before the query sources are known, allowing its construction cost to be amortized across repeated queries on the same graph. The index occupies $\mathrm{O}_{\Delta,r}(N)$ cells and answers each public-source query using $\mathrm{O}_{\Delta,r}(1)$ accesses. For fixed Δ, r, β, the expected construction cost is $\mathrm{O}_{\Delta,r,\beta}(N \log N)$ and becomes linear in N when ε exceeds a sufficiently large constant multiple of $\log N$. We can ensure that all query answers are correct with probability at least $1-\beta$, while the complete trace of construction and any fixed sequence of public-source queries remains pure ε-DO.

1.2 Related work and discussions

Differential obliviousness and composition. Alongside the foundations developed by Chan et al. [4], Mazloom and Gordon [14] study differentially private access patterns in secure computation, including graph-parallel computations such as PageRank. Zhou et al. [22] introduce neighbor-preserving differential obliviousness to address composition: neighboring inputs need not produce neighboring intermediate representations. This issue also arises in graph exploration, where one edge change can affect several later frontiers. Our propagation algorithms account for these changes through explicit bounds on frontier sensitivity.

Access lower bounds and sampling. Patel, Persiano, and Yeo [15] establish access lower bounds and sampling constructions for differentially private information retrieval. Persiano and Yeo [17] develop a broader lower-bound framework based on transferring event probabilities between inputs. Our information-panel argument follows this approach: correctness forces a read on one input, and privacy transfers its probability to another. The graph construction supplies many disjoint panels around every prescribed graph and source. Our two-hop upper bound adapts the retrieval-sampling idea of Patel et al. to the private frontier stored in the source record.

Oblivious graph processing. GraphOS [10] combines trusted hardware with oblivious data structures to support graph traversals and other graph algorithms. GORAM [8] uses secure multiparty computation and an index built using ORAM for queries on federated graphs, including neighbor retrieval, counting, and filtering. Grove [9] supports adjacency access in encrypted graphs through delayed duplication, including multi-hop queries that return vertices within a given distance of a source. These systems provide graph queries over protected representations. We study how the DO privacy parameter affects the access complexity of fixed-radius ball queries from raw padded adjacency records, and include the cost of constructing a protected representation.

Oblivious query processing. Arasu and Kaushik [1] develop oblivious algorithms for relational queries, including joins and aggregation, whose traces depend only on the query and its input and output sizes. Chu et al. [5] show that DO allows two-way joins to avoid padding every execution to the largest possible output size. Wu, Dong, and Hu [20] extend this approach to multi-way joins using private bounds on join size. For fixed Δ, r, our graph queries have bounded output size, yet privacy can still force many input reads. Our preprocessing combines private bucket sampling with fully oblivious batched joins to propagate neighborhoods for all sources.

Preprocessing for private retrieval. Persiano and Yeo [16] and Yeo [21] study trade-offs between preprocessing information and subsequent database accesses in private information retrieval. These works protect the requested index. In our setting the source is public and the graph is private, so we protect the trace of index construction from the raw graph and then answer queries by reading fixed rows. Our preprocessing bounds quantify both the cost of building the index and the cost of subsequent queries.

2 Preliminaries

A *graph* is a pair $G = (V, E)$, where V is the finite set of N vertices and $E \subset \mathcal{P}(V, 2) = \{\{u, v\} : u, v \in V, u \neq v\}$ is a set of undirected edges. For simplicity, we use $V(G)$ and $E(G)$ to denote the vertex set and the edge set of a certain graph G. Throughout the paper, all graphs are finite, simple, undirected, unweighted, connected, and labeled. For any vertex $u \in V$, let $N_G(u) = \{v \in V : \{u, v\} \in E\}$ denote all *neighbors* of u. For every N, let $\mathcal{D}_{N,\Delta}$ denote the domain of all N-vertex graphs satisfying $|N_G(u)| \leq \Delta$ for every $u \in V(G)$.

Given a graph G, we define the *distance* between two vertices as the length of a shortest path between them, denoted by $\mathrm{dist}_G(\cdot, \cdot)$. Notice that when u and v are not connected in G, we have $\mathrm{dist}_G(u, v) = \infty$. Further, the *graph ball* with center $s \in V$ and radius $r \in \mathbb{N}_+$ is a vertex set

$$\mathsf{B}_r(G, s) = \{v \in V : \mathrm{dist}_G(s, v) \leq r\}.$$

To obtain $\mathsf{B}_r(G, s)$, a simple approach is to perform a truncated breadth-first search (BFS), briefly described in Algorithm 1, only incurring $\mathrm{O}_{\Delta,r}(1)$ record accesses. For later use, define the degree-Δ radius-r tree bound

$$M_r := 1 + \Delta \sum_{j=0}^{r-1} (\Delta - 1)^j.$$

Then $|\mathsf{B}_r(G, s)| \leq \min\{N, M_r\}$. Our focus is the local, fixed-radius regime, where M_r is much smaller than N. Once $M_r = \Omega(N)$, this bound reduces to the trivial bound N and no longer captures locality.

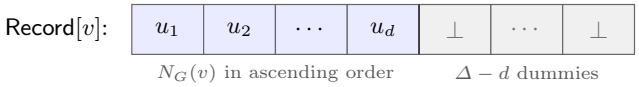
Algorithm 1: GRAPHBALL(G, s, r)

Input: Graph $G = (V, E)$, center $s \in V$, radius $r \in \mathbb{N}_+$.
Output: The graph ball $\mathsf{B}_r(G, s)$.

```
Initialize vertex sets B, F ← {s};
for i ← 1 to r do
    F ← (⋃_{u∈F} N_G(u)) \ B;
    B ← B ∪ F;
return B;
```

2.1 Memory model and observable access traces

Let $f\colon \mathcal{X} \to \mathcal{Y}$ be an algorithm implemented as a RAM program. The *access pattern* $\mathsf{Access}^f(x)$ is the chronological sequence of all physical memory accesses made by f after receiving input $x \in \mathcal{X}$. In our paper, we adopt the cell-probe model, and assume that graphs are stored in a padded adjacency list, i.e., a list of N cell arrays. For each $v \in V$, the record corresponding to v, denoted by $\mathsf{Record}[v]$, consists of Δ cells of $w = \Theta(\log N)$ bits. These cells store its neighbors' labels, with the remaining cells padded with $\perp$. The padding length Δ is public, and genuine labels in each record appear in increasing order. The following diagram visualizes $\mathsf{Record}[v]$ with neighbors $u_1, u_2, \ldots, u_d$.



Apart from the padded adjacency records $\mathsf{Record}[v]$, the machine has a mutable, graph-independently initialized workspace and O(1) private w-bit registers. Inside the machine, each cell-level operation costs O(1) runtime. The addresses and read/write operations of all memory accesses except register accesses are observable to the adversary, in their execution order; memory contents and instruction timing are private. Answers are written to a private write-only stream and interpreted as the set of emitted labels, ignoring order and repetitions.

2.2 Obliviousness and differential obliviousness

Given the observable surface above, *(full) obliviousness* [11] requires the distributions of access patterns induced by different inputs with the same public parameters to be statistically indistinguishable. As a relaxation, *differential obliviousness* [4] degrades the security guarantee to *differential privacy* [6], and requires the access pattern to obey similar distributions for similar inputs. The similarity here is defined by the notion of *neighboring datasets*. For graphs, there are varieties of neighboring relations. We adopt the edge-level neighboring relation here, and provide three different versions, which are commonly-adopted in literature.

Definition 1 (Neighboring graphs). *Given two graphs $G, G' \in \mathcal{D}_{N,\Delta}$ satisfying $V(G) = V(G')$*[1]*.*

1. *The add/delete relation $G \sim_{\text{add}} G'$ holds if $E(G) = E(G') \cup \{e\}$ or $E(G') = E(G) \cup \{e\}$.*
2. *The replacement relation $G \sim_{\text{rep}} G'$ holds if $E(G') = (E(G) \setminus \{e\}) \cup \{f\}$ for $e \in E(G)$ and $f \notin E(G)$.*
3. *The 2-switch relation $G \sim_{\text{sw}} G'$ holds if G' is obtained by a degree-preserving 2-switch, i.e., two vertex-disjoint edges $\{a, b\}, \{c, d\} \in E(G)$ are replaced by two new edges $\{a, c\}, \{b, d\} \notin E(G)$ (or $\{a, d\}, \{b, c\} \notin E(G)$).*

Figure 1 visualizes the three kinds of neighboring relations. Note that all three neighboring relations require $V(G) = V(G')$. We therefore slightly abuse notation and write G for $E(G)$ when describing edge updates.

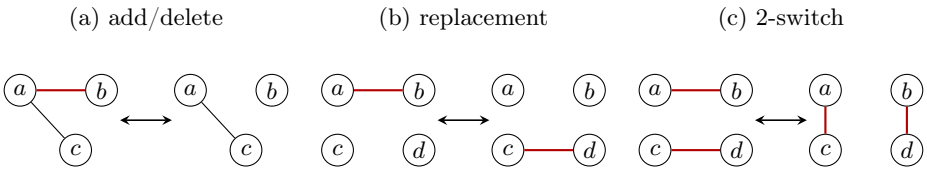


Fig. 1. Examples of the three neighboring relations.

Further, (edge-level) differential obliviousness is defined as follows.

Definition 2 (Differential obliviousness). *All public parameters, including the source s, are fixed in the following comparison and suppressed in the notation $\mathsf{Access}^f(G)$. Given $\varepsilon \in [0, +\infty)$, $\delta \in [0, 1]$, and a neighboring relation $\sim$, a RAM-implemented graph algorithm f with input domain $\mathcal{D}_{N,\Delta}$ is* (ε, δ)-differentially oblivious ((ε, δ)-DO for short) *if for every pair of neighboring graphs $G \sim G'$ and every measurable set $\mathcal{S}$ of finite access patterns,*

$$\mathbb{P}[\mathsf{Access}^f(G) \in \mathcal{S}] \leq e^{\varepsilon} \cdot \mathbb{P}[\mathsf{Access}^f(G') \in \mathcal{S}] + \delta.$$

Following the DP literature, in Definition 2, ε is called the *privacy budget*, quantifying the privacy loss, and δ is the privacy slack. We consider, in particular, the regime where δ is negligibly small with respect to N, denoted by $\delta = \text{negl}(N)$; the general definition allows $\delta \in [0, 1]$. As a privacy metric, smaller values of ε and δ are always more desirable.

2.3 Cost metric

Sections 3–5 consider a single graph ball query with a public source $s \in V$, while Section 6 considers repeated queries on the same graph. For any fixed

[1] Notice that G and G' must be included in $\mathcal{D}_{N,\Delta}$. For example, if G is a Δ-regular graph, $G' = G \cup \{e\}$ will not be regarded as its legal neighbor, since $G' \notin \mathcal{D}_{N,\Delta}$.

algorithm, its code, initial workspace and register contents, and distribution of random coins are the same for every input graph. Thus, the algorithm obtains graph-dependent information only by reading the padded (with length Δ) adjacency list. Any graph-dependent preprocessing is part of the execution, and all observable accesses made before the answer is returned are included in the access pattern and charged.

For any RAM-implemented graph algorithm f, let $T^f_{\mathrm{read}}(G)$ be the number of cells read from the padded adjacency list on receiving input G, and let $T^f_{\mathrm{work}}(G)$ be the number of reads and writes to workspace cells. We measure the total observable access cost by

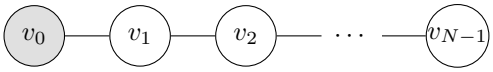
$$T^f_{\mathrm{total}}(G) := T^f_{\mathrm{read}}(G) + T^f_{\mathrm{work}}(G).$$

In particular, reading an entire padded record costs Δ input accesses. All expectations are over the algorithm's internal randomness.

3 Warm-up: Loss of Locality on a Path

In this section, we take a path graph as an example, to illustrate how differential obliviousness destroys locality. Although it is evident that an ordinary constant-hop ball query on a path has constant access cost, every (ε, δ)-DO algorithm with error bounded $\beta < 1/2$ requires $\Omega(Ne^{-\varepsilon})$ expected input reads for sufficiently small δ, under the add/delete neighboring relation.

Now, consider the r-hop ball query $\mathsf{B}_r(H_N, s)$ on path graph H_N with $N \geq 7$ vertices, where the vertices are indexed from 0 to $N-1$, visualized as follows. Let $s = v_0$ be the (public) query center, which is shaded in the diagram.



When $r = 1$, since the center vertex s is public, we can access $\mathsf{Record}[s]$ directly. Noting that this process is just a linear scan on the corresponding array, it satisfies $(0, 0)$-DO, requiring $\Theta(\Delta) = \mathrm{O}_{\Delta,r}(1)$ input-cell reads.

When $r \geq 2$, however, the natural local traversal is no longer oblivious, as its access pattern of the second hop depends on the contents of $\mathsf{Record}[s]$. As shown below, already for $r = 2$, hiding this dependence requires $\Omega_\beta(Ne^{-\varepsilon})$ expected input reads under (ε, δ)-DO with sufficiently small δ.

In the rest of this section, we prove a lower bound for 2-hop graph ball algorithms satisfying (ε, δ)-DO under the add/delete neighboring relation, with error probability at most β. We form $m = \lfloor (N-3)/4 \rfloor$ disjoint panels of four consecutive vertices along the path, starting at v_2, and use correctness and privacy to lower-bound the probability of reading each panel.

Add/delete neighboring relation. For $i = 0, 1, \ldots, m-1$, we define the i-th panel $P_i = \{a_i, z_i, b_i, c_i\}$, where

$$a_i = v_{4i+2},\ z_i = v_{4i+3},\ b_i = v_{4i+4},\ c_i = v_{4i+5}.$$

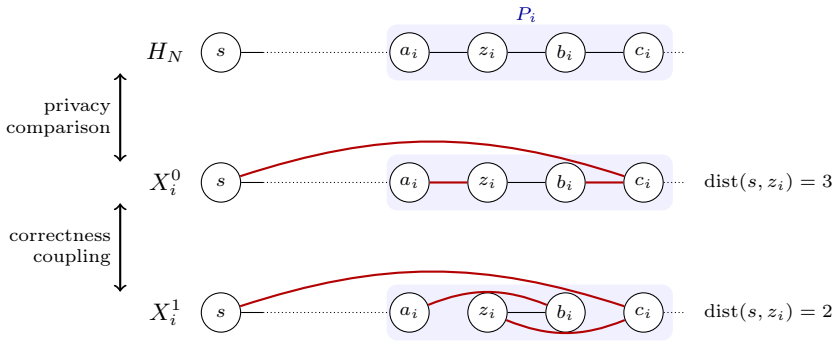


Fig. 2. The add/delete construction. Each row shows only edges present in that input.

It is easy to check that these panels are disjoint and consecutive in H_N. Now, we can construct two graphs X_i^0, X_i^1 as follows. Here, X_i^0 satisfies $H_N \sim_{\mathrm{add}} X_i^0$ and activates the panel P_i by adding the edge $\{s, c_i\}$, while X_i^1 differs from X_i^0 only by a 2-switch inside the panel P_i.

$$X_i^0 = H_N \cup \big\{\{s, c_i\}\big\},$$
$$X_i^1 = \Big(X_i^0 \setminus \big\{\{a_i, z_i\}, \{b_i, c_i\}\big\}\Big) \cup \big\{\{a_i, b_i\}, \{z_i, c_i\}\big\}.$$

Both inputs are connected, with degree upper bounded by 3. Moreover, we have

$$\mathrm{dist}_{X_i^0}(s, z_i) = 3 \quad \text{and} \quad \mathrm{dist}_{X_i^1}(s, z_i) = 2,$$

which indicates that $z_i \notin \mathsf{B}_2(X_i^0, s)$ and $z_i \in \mathsf{B}_2(X_i^1, s)$. In fact, in X_i^0 the two path routes from s to z_i have lengths $4i+3$ and 3, while in X_i^1 the route s, c_i, z_i has length two and $\{s, z_i\}$ is absent. Figure 2 illustrates the construction.

Now, we are ready to describe the loss of locality for graph ball query on H_N under the add/delete relation. Here, we define the privacy transfer factor

$$\Lambda(\varepsilon, \delta, \beta) := \max\big\{0, e^{-\varepsilon}(1 - 2\beta - \delta), 1 - 2\beta e^{\varepsilon} - \delta\big\}.$$

In particular, when $\beta = 0$, we have $\Lambda(\varepsilon, \delta, 0) = 1 - \delta$.

Proposition 1 (Loss of locality on a path under $\sim_{\mathbf{add}}$). *Let $N \geq 7$, $\varepsilon \geq 0$, $0 \leq \delta \leq 1$, and $0 \leq \beta < 1/2$. Suppose that an algorithm outputs $\mathsf{B}_2(G, s)$ with error probability at most β throughout $\mathcal{D}_{N,\Delta}$. If it is (ε, δ)-DO under the add/delete neighboring relation, then*

$$\mathbb{E}[T_{\mathrm{read}}(H_N, s)] \geq m \cdot \Lambda(\varepsilon, \delta, \beta).$$

The same bound holds for $\mathbb{E}[T_{\mathrm{total}}(H_N, s)]$.

Proof. For a vertex set P, let $\mathsf{Hit}(P)$ denote the event that the algorithm reads a cell of $\mathsf{Record}[v]$ for some $v \in P$. Fix $i \in \{0, 1, \ldots, m-1\}$ and couple the

executions on X_i^0 and X_i^1 using the same random coins. Their initial states agree, and their input records agree outside P_i. Thus, until either execution reads P_i, the two executions receive the same values and make the same subsequent access decisions. Consequently, they either hit P_i at the same step or both terminate without hitting it. Let q_i be this common hit probability.

For $b \in \{0, 1\}$, let C_b be the event that the execution on X_i^b is correct. On any coupled run that does not hit P_i, the two executions produce the same output, whereas their correct answers differ. Hence at most one of C_0 and C_1 occurs on such a run, and therefore

$$\mathbb{P}[C_0] + \mathbb{P}[C_1] \leq 1 + \mathbb{P}[\mathsf{Hit}(P_i)] = 1 + q_i.$$

Using correctness on both inputs, we have $2(1 - \beta) \leq 1 + q_i$, i.e., $q_i \geq 1 - 2\beta$.

Now let $p_i = \mathbb{P}_{H_N}[\mathsf{Hit}(P_i)]$. Since $H_N \sim_{\mathrm{add}} X_i^0$ and the hit and no-hit events are determined by the observable trace, DO in the two directions indicates

$$q_i \leq e^{\varepsilon} p_i + \delta \quad \text{and} \quad 1 - p_i \leq e^{\varepsilon}(1 - q_i) + \delta.$$

Combining these inequalities with $q_i \geq 1 - 2\beta$, we have

$$p_i \geq e^{-\varepsilon}(1 - 2\beta - \delta) \quad \text{and} \quad p_i \geq 1 - 2\beta e^{\varepsilon} - \delta.$$

Together with $p_i \geq 0$, this shows that $p_i \geq \Lambda(\varepsilon, \delta, \beta)$.

Finally, as the panels are disjoint, we obtain the claimed read bound

$$\mathbb{E}[T_{\mathrm{read}}(H_N, s)] \geq \sum_{i=0}^{m-1} \mathbb{P}_{H_N}[\mathsf{Hit}(P_i)] = \sum_{i=0}^{m-1} p_i \geq m\Lambda(\varepsilon, \delta, \beta).$$

The total-cost bound follows from $T_{\mathrm{total}} \geq T_{\mathrm{read}}$, which completes the proof. □

4 Universal Lower Bounds via Information Panels

This section extends the proof idea on path graphs in Section 3 to general bounded-degree graphs.

4.1 The Information-Panel Method

We now abstract the argument of Section 3. The probability transfer follows the approach of Persiano and Yeo [17] and is closely related to the access lower bound for differentially private information retrieval [15].

First of all, we abstract the bundles consisting of 4 vertices, which have been repeatedly used in Section 3, as *information panel*.

Definition 3 (Information panel). *Given a graph query Q and its public parameters, two input graphs X^0, X^1 with the same vertex set V have an* information panel $P \subseteq V$ *if*

1. *their padded adjacency records agree for every vertex outside P, and*
2. $Q(X^0) \neq Q(X^1)$.

Further, let $\mathsf{Hit}(P)$ *denote the event that the execution of the RAM-implemented algorithm reads any cell of* $\mathsf{Record}[v]$ *for some* $v \in P$.

Then we have the following lemma.

Lemma 1. *Given* $\varepsilon \geq 0$, $0 \leq \delta \leq 1$, *and* $0 \leq \beta < 1/2$, *let* A *be an* (ε, δ)*-DO algorithm for a graph query* Q *under a neighboring relation* $\sim$ *on* $\mathcal{D}_{N,\Delta}$, *with error probability at most* β *on every input. For any* $H \in \mathcal{D}_{N,\Delta}$, *suppose there is a family of pairwise disjoint vertex sets* $\mathcal{P}$ *such that, for every* $P \in \mathcal{P}$, *there exist* $X_P^0, X_P^1 \in \mathcal{D}_{N,\Delta}$ *for which* P *is an information panel and* $H \sim X_P^0$. *Then*

$$\mathbb{E}[T_{\mathrm{read}}(H)] \geq |\mathcal{P}| \cdot \Lambda(\varepsilon, \delta, \beta).$$

The same expression also lower bounds $\mathbb{E}[T_{\mathrm{total}}(H)]$.

Proof. Fix $P \in \mathcal{P}$ and couple the executions on X_P^0 and X_P^1 using the same random coins. Their initial registers and workspace agree because the initial state is graph independent. Until the first read from P, every input value returned in the two executions is identical. Inductively, their control state, registers, workspace contents, and next access are therefore identical. They either hit P at the same first cell or both stop without hitting it. Write q_P for this common hit probability.

Let C_b denote the event that the execution on X_P^b is correct. If neither execution hits P, they return the same value and at most one is correct. If the panel is read, both may be correct. Thus,

$$\mathbb{P}[C_0] + \mathbb{P}[C_1] \leq 1 + \mathbb{P}[\mathsf{Hit}(P)] = 1 + q_P.$$

Correctness on both inputs hence gives $q_P \geq 1 - 2\beta$.

Write $p_P = \mathbb{P}_H[\mathsf{Hit}(P)]$. Privacy for the hit event gives

$$q_P \leq e^{\varepsilon} p_P + \delta, \qquad p_P \geq e^{-\varepsilon}(1 - 2\beta - \delta).$$

Applying privacy in the reverse direction to the no-hit event gives

$$1 - p_P \leq e^{\varepsilon}(1 - q_P) + \delta \leq 2\beta e^{\varepsilon} + \delta,$$

and therefore

$$p_P \geq 1 - 2\beta e^{\varepsilon} - \delta.$$

Together with $p_P \geq 0$, these inequalities give $p_P \geq \Lambda(\varepsilon, \delta, \beta)$.

Since the panels are disjoint, each input-cell read can hit at most one of them,

$$\mathbb{E}[T_{\mathrm{read}}(H)] \geq \sum_{P \in \mathcal{P}} \mathbb{P}_H[\mathsf{Hit}(P)] = \sum_{P \in \mathcal{P}} p_P \geq |\mathcal{P}| \cdot \Lambda(\varepsilon, \delta, \beta),$$

which completes the proof. □

At zero error, the reverse no-hit inequality is essential. Under pure DO, every panel must be read with probability one for every finite ε. We next construct linearly many disjoint panels for every sufficiently large connected bounded-degree graph and every source.

4.2 Universal BFS Construction

We now extend the path construction to general connected bounded-degree graphs. In brief, for any center $s \in G$, we consider the BFS tree of which the root is s. Then, we construct $\Theta(N)$ distinct subgraphs of 4 vertices in the tree. By selecting $\Theta(N)$ panels in this subgraph family, we prove the lower bound via Lemma 1. Figure 3 summarizes this workflow.

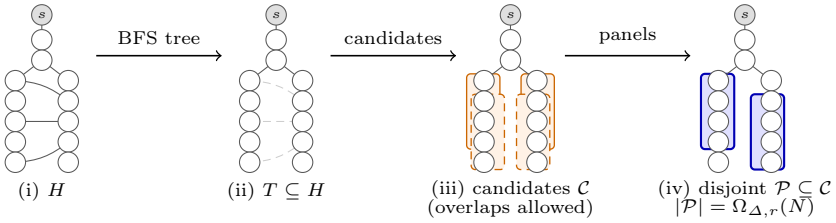


Fig. 3. Workflow of the universal BFS panel construction.

Our main theorem on the lower bound is stated as follows.

Theorem 1 (Universal lower bound). *Given $\Delta, r \geq 2$ and neighboring relation $\sim \in \{\sim_{\mathrm{add}}, \sim_{\mathrm{rep}}, \sim_{\mathrm{sw}}\}$ on $\mathcal{D}_{N,\Delta}$, there exist constants $c_{\Delta,r} > 0$ and $N_0(\Delta, r) \in \mathbb{N}$ such that for every $N \geq N_0(\Delta, r)$, any (ε, δ)-DO algorithm A for radius-r graph ball queries under $\sim$, with error probability at most β, must satisfy*

$$\mathbb{E}[T^A_{\mathrm{total}}(H, s)] \geq \mathbb{E}[T^A_{\mathrm{read}}(H, s)] \geq c_{\Delta,r} N \cdot \Lambda(\varepsilon, \delta, \beta), \quad \text{for all } (H, s),$$

where $\varepsilon \geq 0$, $0 \leq \delta \leq 1$, and $0 \leq \beta < 1/2$.

Proof. Fix $\sim$ and (H, s), and choose an s-rooted BFS spanning tree T of H. For each candidate panel P_c, we construct an anchor X^0_c neighboring H and a witness X^1_c whose records agree with those of X^0_c outside P_c but whose radius-r ball around s differs from that of X^0_c, as required by Lemma 1.

Common panel construction. Write $\ell(v) = \mathrm{dist}_H(s, v)$, and let $\pi(v)$ be the parent of v in T. For a center c of depth $d = \ell(c) \geq 2r + 1$, define $p^j_c = \pi^{r-2+j}(c)$ for $j = 0, 1, 2, 3$ and $P_c = \{p^0_c, p^1_c, p^2_c, p^3_c\}$. These four vertices have depths $d - r + 2$, $d - r + 1$, $d - r$, and $d - r - 1$, respectively. In each construction below, p^2_c has distance $r + 1$ from s in X^0_c, and the panel switch creates a path of length r from s to p^2_c in X^1_c. The endpoints of every edge of H differ in depth by at most one. Therefore $\{p^0_c, p^2_c\}$ and $\{p^1_c, p^3_c\}$, whose endpoints differ in depth by two, are nonedges. All panel vertices have depths between r and d. All three constructions use the following 2-switch to define X^1_c from X^0_c:

$$X^1_c = X^0_c \setminus \big\{\{p^0_c, p^1_c\}, \{p^2_c, p^3_c\}\big\} \cup \big\{\{p^0_c, p^2_c\}, \{p^1_c, p^3_c\}\big\}. \tag{1}$$

Only edges with both endpoints in P_c change, so the padded adjacency records of X^0_c and X^1_c agree outside P_c. Each panel vertex loses one edge and gains one edge. We check below that both graphs are simple and connected.

Add/delete activation. Set $X_c^0 = H \cup \big\{\{s,c\}\big\}$. Since $d > 1$, we have $\{s,c\} \notin E(H)$, so $H \sim_{\text{add}} X_c^0$. Since $s \notin P_c$, adding $\{s,c\}$ does not change any edge within P_c. Thus X_c^1 is simple. Deleting the two panel edges from T leaves three components containing p_c^0, $\{p_c^1, p_c^2\}$, and p_c^3, respectively. The two new panel edges connect the first component to the second and the second to the third. Thus both X_c^0 and X_c^1 are connected.

Let $p = p_c^2$, whose depth is $d - r$. A shortest s–p path in X_c^0 either avoids $\{s,c\}$ and has length at least $d-r$, or starts with $\{s,c\}$ and then uses at least r edges of H. Both bounds are attained: follow T from s to p, or take $\{s,c\}$ and then follow T from c to p. Thus

$$\operatorname{dist}_{X_c^0}(s,p) = \min\{d-r, r+1\} = r+1.$$

In X_c^1, the path $s - c - \cdots - p_c^0 - p_c^2$ has length $1 + (r-2) + 1 = r$. Hence p_c^2 lies outside the ball in X_c^0 and inside the ball in X_c^1, and P_c is an information panel.

Replacement activation. Let x be the neighbor of s on the s–c path in T, and

$$X_c^0 = H \setminus \{\{s,x\}\} \cup \{\{s,c\}\}.$$

Since $\{s,x\} \in E(H)$ and $\{s,c\} \notin E(H)$, we have $H \sim_{\text{rep}} X_c^0$. Deleting $\{s,x\}$ from T separates s and c, and adding $\{s,c\}$ reconnects the two components. Thus $T_c := T \setminus \{\{s,x\}\} \cup \{\{s,c\}\}$ is a spanning tree of X_c^0.

Let X_c^1 be the graph defined by (1). Since $s, x \notin P_c$, constructing X_c^0 does not change any edge within P_c. Thus X_c^1 is simple, and T_c still contains the four-vertex panel path. As in the add/delete case, the two new panel edges reconnect the three components, so X_c^1 is connected. The switch preserves degrees, and the records of X_c^0 and X_c^1 agree outside P_c.

Let $p = p_c^2$. The add/delete distance calculation gives $\operatorname{dist}_{H \cup \{\{s,c\}\}}(s,p) = r+1$. Deleting $\{s,x\}$ cannot shorten this distance, and the length-$(r+1)$ path from s through c to p avoids the deleted edge. Thus

$$\operatorname{dist}_{X_c^0}(s,p) = r+1.$$

In X_c^1, the path $s - c - \cdots - p_c^0 - p_c^2$ has length $1 + (r-2) + 1 = r$. Hence p_c^2 lies outside the ball in X_c^0 and inside the ball in X_c^1, and P_c is an information panel.

Degree-preserving 2-switch activation. Restrict to centers c with a child a in T, and let x be the neighbor of s on the s–c path in T. Define

$$X_c^0 = H \setminus \big\{\{s,x\},\{c,a\}\big\} \cup \big\{\{s,c\},\{x,a\}\big\}. \tag{2}$$

The four endpoints are distinct. Both new edges are absent from H because their endpoints differ in depth by d. The graph $T \setminus \big\{\{s,x\},\{c,a\}\big\} \cup \big\{\{s,c\},\{x,a\}\big\}$ is a spanning tree of X_c^0: the two new edges reconnect the component containing s, the component containing x, c, and the subtree rooted at a. Thus X_c^0 is connected, and (2) is a valid degree-preserving 2-switch.

The depths of s, x, a are $0, 1, d+1$, so none of these vertices belongs to P_c. Every edge changed in constructing X_c^0 therefore has an endpoint outside P_c. Hence the two switches share no edge, even when $r = 2$ and $c = p_c^0$, and X_c^1 is simple. The spanning tree above still contains the four-vertex panel path, so the same argument as in the add/delete case shows that X_c^1 is connected. Both switches preserve every vertex's degree, so X_c^1 has the same labeled degree vector and edge count as H.

To prove that $\mathrm{dist}_{X_c^0}(s, p_c^2) = r + 1$, we must also account for paths that use the new edge $\{x, a\}$. Let $H^- = H \setminus \big\{\{s, x\}, \{c, a\}\big\}$ and $p = p_c^2$. Every path from u to v in H^- has length at least $|\ell(u) - \ell(v)|$. Consider a shortest s–p path Q in X_c^0. If Q uses $\{s, c\}$, it must start with this edge, traversed from s to c.

(i) If Q avoids both new edges, then $|Q| \geq d - r$.

(ii) If Q uses $\{x, a\}$ but not $\{s, c\}$, traversing $\{x, a\}$ from x to a gives

$$|Q| \geq \ell(x) + 1 + |\ell(a) - \ell(p)| = r + 3,$$

whereas traversing it from a to x gives

$$|Q| \geq \ell(a) + 1 + |\ell(p) - \ell(x)| = 2d - r + 1.$$

(iii) If Q uses $\{s, c\}$ but not $\{x, a\}$, then

$$|Q| \geq 1 + |\ell(c) - \ell(p)| = r + 1.$$

(iv) If Q uses both new edges, it starts with $\{s, c\}$. Traversing $\{x, a\}$ as $a \to x$ or $x \to a$ then gives, respectively,

$$|Q| \geq 1 + |\ell(c) - \ell(a)| + 1 + |\ell(x) - \ell(p)| = d - r + 2,$$
$$|Q| \geq 1 + |\ell(c) - \ell(x)| + 1 + |\ell(a) - \ell(p)| = d + r + 2.$$

Since $d \geq 2r + 1$, we have $|Q| \geq r + 1$ in every case. The path that starts with $\{s, c\}$ and then follows T from c to p uses neither deleted edge and has length $r + 1$, so $\mathrm{dist}_{X_c^0}(s, p_c^2) = r + 1$. In X_c^1, the path $s - c - \cdots - p_c^0 - p_c^2$ has length r. Hence p_c^2 lies outside the ball in X_c^0 and inside the ball in X_c^1, and P_c is an information panel.

Disjoint panel selection. Let

$$R_{\Delta,r} := \sum_{j=r-2}^{r+1} (\Delta - 1)^j \quad \text{and} \quad S_{\Delta,r} := 1 + \Delta \sum_{j=0}^{2r-1} (\Delta - 1)^j.$$

Then $S_{\Delta,r}$ bounds the number of vertices at BFS depth at most $2r$, while $R_{\Delta,r}$ bounds the number of candidate panels containing any fixed vertex. Hence at least $N - S_{\Delta,r}$ centers are available for the add/delete and replacement activations. For the switch construction, let I be the number of nonleaves in the rooted BFS tree. If $N = 1$, the switch bound is vacuous. Otherwise the root is a nonleaf, and counting children gives

$$N - 1 \leq \Delta + (I - 1)(\Delta - 1) = (\Delta - 1)I + 1,$$

and therefore $I \geq (N-2)/(\Delta-1)$. After excluding vertices at depth at most $2r$, at least $(N-2)/(\Delta-1) - S_{\Delta,r}$ eligible nonleaves remain.

A vertex v belongs to P_c only if c is a descendant of v in T at distance $r-2$, $r-1$, r, or $r+1$. Since no panel contains the root, each vertex belongs to at most $R_{\Delta,r}$ candidate panels. Selecting a panel and discarding all candidates that intersect it removes at most $4R_{\Delta,r}$ candidates. Hence greedy selection produces pairwise disjoint panel families of sizes at least

$$\frac{(N - S_{\Delta,r})_+}{4R_{\Delta,r}} \quad \text{and} \quad \frac{1}{4R_{\Delta,r}} \left(\frac{N-2}{\Delta-1} - S_{\Delta,r} \right)_+ .$$

Here $(x)_+ = \max\{x, 0\}$. The first bound applies to both add/delete and replacement activation, and the second to 2-switch activation.

For each selected panel, we have $H \sim X_c^0$, and the two inputs X_c^0, X_c^1 belong to $\mathcal{D}_{N,\Delta}$ and give different answers. No neighboring relation between X_c^0 and X_c^1 is required. Applying Lemma 1 gives the explicit bounds

$$\mathbb{E}[T_{\mathrm{read}}(H,s)] \geq \frac{(N - S_{\Delta,r})_+}{4R_{\Delta,r}} \Lambda(\varepsilon,\delta,\beta),$$
$$\mathbb{E}[T_{\mathrm{read}}(H,s)] \geq \frac{1}{4R_{\Delta,r}} \left(\frac{N-2}{\Delta-1} - S_{\Delta,r} \right)_+ \Lambda(\varepsilon,\delta,\beta),$$

for $\sim \in \{\sim_{\mathrm{add}}, \sim_{\mathrm{rep}}\}$ and $\sim = \sim_{\mathrm{sw}}$, respectively.

Letting $N_0 = 4 + 2(\Delta-1)S_{\Delta,r}$ and $c_{\Delta,r} = \frac{1}{8(\Delta-1)R_{\Delta,r}}$, both coefficients of $\Lambda(\varepsilon,\delta,\beta)$ are at least $c_{\Delta,r}N$ whenever $N \geq N_0$, which completes the proof. □

The bound is pointwise at the prescribed pair (H,s). By contrast, ordinary exploration scans only the records of vertices at distance less than r, at most $1 + \Delta\sum_{j=0}^{r-2}(\Delta-1)^j$ records, and thus costs $\mathrm{O}_{\Delta,r}(1)$ cell reads.

For zero error, $\Lambda(\varepsilon,\delta,0) = 1-\delta$, Theorem 1 gives an $\Omega_{\Delta,r}(N(1-\delta))$ lower bound. In particular, under pure DO, the expected cost remains $\Omega_{\Delta,r}(N)$ for every finite ε (even if it is non-constant). Further, when we allow positive error, non-constant ε can lead to lower time complexity. Formally, we have the following corollary, which indicates that

Corollary 1 (Positive-error lower bound). *Given $\Delta, r \geq 2$, $\varepsilon \geq 0$, $0 \leq \delta \leq (1-2\beta)/2$, $0 < \beta < 1/2$, and a neighboring relation $\sim \in \{\sim_{\mathrm{add}}, \sim_{\mathrm{rep}}, \sim_{\mathrm{sw}}\}$ on $\mathcal{D}_{N,\Delta}$, there exist constants $c_{\Delta,r,\beta} > 0$ and $N_0(\Delta,r) \in \mathbb{N}$ such that for every $N \geq N_0(\Delta,r)$, any (ε,δ)-DO algorithm A for radius-r graph ball queries under $\sim$, with error probability at most β, must satisfy*

$$\mathbb{E}[T^A_{\mathrm{total}}(H,s)] \geq \mathbb{E}[T^A_{\mathrm{read}}(H,s)] \geq c_{\Delta,r,\beta}\big(1 + Ne^{-\varepsilon}\big), \quad \textit{for all } (H,s).$$

5 Upper Bounds via Sampling and Propagation

This section gives two upper-bound constructions for fixed-radius graph ball queries. We first give a fully oblivious baseline that propagates the discovered

ball through graph-independent scans. We then replace these full scans by biased sampling, obtaining a privacy-dependent improvement for every fixed radius and a tight bound for radius 2.

5.1 Baseline: A naive scan-propagation approach

As a baseline, Algorithm 2 scans all vertices in public order for each new layer.

Algorithm 2: FullScanBall(G, s, r)

Input: Graph $G = (V, E)$, center $s \in V$, radius r.
Output: A vertex array A, whose set of valid entries is exactly $\mathsf{B}_r(G, s)$.

Initialize an M_r-slot buffer $A \leftarrow \{(s, 0), \perp, \perp, \ldots, \perp\}$;
for $j \leftarrow 1$ **to** Δ **do**
  Replace the first $\perp$ in A with $(\mathsf{Record}[s][j], 1)$ if $\mathsf{Record}[s][j]$ is valid;
for $i \leftarrow 1$ **to** $r - 1$ **do**
  for $v \leftarrow 1$ **to** N **do**
    for $j \leftarrow 1$ **to** Δ **do**
      Scan and rewrite A, replacing its first $\perp$ with $(\mathsf{Record}[v][j], i + 1)$ iff $(v, i) \in A$ and $\mathsf{Record}[v][j]$ is a valid label not already stored in A;
return All vertices included in A;

Since all item-level operations are completed via linear scans, FullScanBall(G, s, r) satisfies $(0, 0)$-DO. Formally, we have the following proposition.

Proposition 2. *Given $N \geq 1$ and $r \geq 2$, there exists a deterministic algorithm that is $(0, 0)$-DO under all three neighboring relations and, for every $G \in \mathcal{D}_{N,\Delta}$ and public source $s \in V(G)$, outputs $\mathsf{B}_r(G, s)$ correctly, with both T_{read} and T_{total} bounded by $\mathrm{O}_{\Delta,r}(N)$.*

Proof. Consider FullScanBall(G, s, r) described in Algorithm 2. The algorithm keeps the valid entries of A packed before its dummy entries and stores each vertex at most once, together with the layer in which it is first discovered. Each replacement is implemented by one complete scan and rewrite of the M_r slots. Upon reaching the first dummy, the scan has already determined whether the required pair occurs and whether the candidate label is already stored; it then either performs the replacement or rewrites the dummy unchanged. Only the current labels, loop counters, and a constant number of Boolean flags are retained in private registers.

After initialization, the valid entries of A are $(s, 0)$ and $(u, 1)$ for $u \in N_G(s)$. Suppose that at the start of round i, A contains precisely $(u, \mathrm{dist}_G(s, u))$ for $u \in \mathsf{B}_i(G, s)$. The round expands every vertex tagged i and inserts each previously unseen valid neighbor with tag $i + 1$. It therefore leaves A containing precisely the vertices of $\mathsf{B}_{i+1}(G, s)$ with their distances from s. Moreover, throughout the

round the stored labels are distinct vertices of $\mathsf{B}_{i+1}(G,s)$. Since $|\mathsf{B}_{i+1}(G,s)| \le M_r$, every required insertion finds a dummy slot. Induction proves correctness after round $r-1$.

The source phase reads the Δ cells of $\mathsf{Record}[s]$, and every later round reads all $N\Delta$ record cells, indicating that $T_{\mathrm{read}} = \Delta(1+(r-1)N) = \mathrm{O}_{\Delta,r}(N)$. Every such read is accompanied by a complete scan and rewrite of A; initialization and output add only $\mathrm{O}(M_r)$ accesses. This shows that $T_{\mathrm{total}} = \mathrm{O}(M_r T_{\mathrm{read}}) = \mathrm{O}_{\Delta,r}(N)$. All loop bounds, input addresses, and workspace addresses depend only on the public parameters and source, and every conditional replacement uses the same read–write schedule. Hence the complete trace is graph independent and, in particular, $(0,0)$-DO under all three neighboring relations. □

5.2 Sampled Propagation for Fixed-Radius Balls

The fully oblivious baseline scans all vertices in every propagation round, even though only a small set needs to be expanded. We now use differential obliviousness to reduce this cost through biased sampling. Algorithm 3, SampledBall, samples discovered vertices with probability p and all other vertices with probability $q \le p$. Every sampled vertex triggers a record scan, but only discovered vertices contribute new neighbors. The parameter p is chosen so that, with probability at least $1-\beta$, every propagation round samples its entire discovered set. To choose q, we allocate the privacy budget across rounds according to the sensitivity of the discovered sets.

The sampling step in Algorithm 3 can be implemented by generating the labels of S_i in increasing order directly from the buffer A, without storing S_i. The following lemma gives such a procedure using only a constant number of private registers, with one array scan per generated label and one final scan.

Lemma 2. *Consider an array of k cells, where k is public. Its first $m = |D|$ cells store the labels of a set $D \subseteq \{1,2,\dots,N\}$ in arbitrary order. For any $0 \le q \le p \le 1$, there exists an algorithm that samples a set $S \subseteq \{1,2,\dots,N\}$ by including each vertex independently with probability p in D and q outside D, incurring $\mathrm{O}(k(|S|+1))$ array accesses.*

Proof. If $q = 1$, then $S = \{1,2,\dots,N\}$. A complete array scan determines the membership bit before each label is returned, and one further scan precedes termination. For $q < 1$, let $t = \frac{p-q}{1-q}$. We generate $S = Q \cup R$, where $Q \subseteq \{1,2,\dots,N\}$ and $R \subseteq D$ are sampled independently with inclusion probabilities q and t, respectively. Both sets are enumerated in increasing order. Their union has independent inclusion events with the required probabilities.

Suppose u is the most recently generated element of R. Conditional on the elements generated so far, the inclusion indicators of larger labels in D remain independent Bernoulli-t. Hence the remaining elements of R can be generated lazily in subsequent scans. A complete array scan suffices to find the next selected label: among the first m entries, it retains the smallest label above the previous one whose independent t-coin succeeds, or determines that none remains.

Algorithm 3: SAMPLEDBALL$(G, s, r, \varepsilon, \beta, \sim)$

Input: Graph $G = (V, E) \in \mathcal{D}_{N,\Delta}$, center $s \in V$, integer radius $r \geq 2$, privacy budget $\varepsilon \in [0, +\infty)$, error probability cap $0 < \beta < 1$, public neighboring relation $\sim \in \{\sim_{\text{add}}, \sim_{\text{rep}}, \sim_{\text{sw}}\}$.

Output: A vertex array A, whose valid entries are exactly $\mathsf{B}_r(G, s)$ with probability at least $1 - \beta$.

Initialize the parameters:

$$\kappa \leftarrow \begin{cases} 2, & \sim = \sim_{\text{sw}}, \\ 1, & \sim \in \{\sim_{\text{add}}, \sim_{\text{rep}}\}, \end{cases} \qquad \sigma_i \leftarrow \begin{cases} 1, & i = 1, \\ \min\{M_i, (\Delta+1)\sigma_{i-1} + \kappa\}, & i \geq 2, \end{cases}$$

$$C^{\sim}_{\Delta,r} \leftarrow \sum_{i=1}^{r-1} \sigma_i, \quad \eta \leftarrow \frac{\varepsilon}{C^{\sim}_{\Delta,r}}, \quad p \leftarrow (1-\beta)^{\frac{1}{\sum_{i=1}^{r-1} M_i}}, \quad q \leftarrow \frac{p}{p + e^{\eta} \cdot (1-p)};$$

Initialize an M_r-slot buffer $A \leftarrow \{s, \perp, \perp, \ldots, \perp\}$;
for $j \leftarrow 1$ **to** Δ **do**
    By a linear scan of A, replace its first $\perp$ with $\mathsf{Record}[s][j]$ if $\mathsf{Record}[s][j]$ is not a $\perp$;
for $i \leftarrow 1$ **to** $r - 1$ **do**
    $A_i \leftarrow$ non-dummy entries in A;
    Sample $S_i \subseteq \{1, 2, \ldots, N\}$ with independent inclusion events, where $\mathbb{P}[v \in S_i] = \begin{cases} p, & v \in A_i, \\ q, & v \notin A_i, \end{cases}$ for all $v \in V$;
    foreach $v \in S_i$ *in increasing order* **do**
        for $j \leftarrow 1$ **to** Δ **do**
            $x \leftarrow \mathsf{Record}[v][j]$;
            if $v \in A_i$ **then**
                Replace the first $\perp$ in A with x iff $x \neq \perp$ and $x \notin A$;
            else
                Scan and rewrite all of A without modifying it;
return A;

Merging the two increasing enumerations and emitting coincident labels only once gives the required order. Membership in D is known for labels from R and can be tested for labels from Q in the same scan used to generate the next label from R. To keep the per-output access pattern uniform, we perform one complete scan for every emitted label, even when no new label from R is needed, and one additional scan before termination.

The merge uses a constant number of private registers to hold the next label from each enumeration, the last label returned from R, and the required counters and flags. Generating Q requires no data-memory accesses, since it uses only public labels and fresh coins within registers. Thus, the total number of array accesses is $\mathrm{O}(k(|S| + 1))$. □

When applying the sampling algorithm of Lemma 2 in round i of Algorithm 3, we have $D = A_i$, $m = |A_i|$, $k = M_r$, and $S = S_i$. The array is the buffer A, whose first m cells contain the discovered labels and remain fixed during the round while new labels are appended. A full scan determines m, and the lemma then generates S_i with $\mathrm{O}(M_r(|S_i| + 1))$ buffer accesses.

Intuitively, sampling vertices outside A_i as well as those inside it limits what the accessed labels reveal about the discovered set. The choice of q ensures that a small change in A_i changes the probability of any sampled set by only a bounded factor. The following lemma quantifies this effect, providing the sampling guarantee needed for the DO proof of the full algorithm.

Lemma 3 (Biased-subset likelihood ratio). *Fix $0 < p < 1$ and $\eta \geq 0$. For a set $A \subseteq [N]$, let μ_A be the product distribution with inclusion probabilities p on A and q outside A, where $q = p/(p + e^{\eta}(1 - p))$. For any $A, A', S \subseteq [N]$, put $d = \max\{|A \setminus A'|, |A' \setminus A|\}$. Then*

$$e^{-\eta d} \leq \frac{\mu_A(S)}{\mu_{A'}(S)} \leq e^{\eta d}.$$

Proof. Let $Z = p + e^{\eta}(1 - p)$, so $1 \leq Z \leq e^{\eta}$. A label in $A \setminus A'$ contributes $\frac{p}{q} = Z$ to the likelihood ratio when sampled and $\frac{1-p}{1-q} = Ze^{-\eta}$ otherwise, in either case at most Z. For a label in $A' \setminus A$, the contribution is at most $\frac{e^{\eta}}{Z}$, and all other labels contribute 1. With $a = |A \setminus A'|$ and $b = |A' \setminus A|$, independence gives

$$\frac{\mu_A(S)}{\mu_{A'}(S)} \leq e^{b\eta} Z^{a-b} \leq e^{\eta \max\{a,b\}}.$$

The lower bound follows by exchanging A and A'. □

We can now establish the correctness, privacy, and access complexity of Algorithm 3. Formally, we have the following theorem.

Theorem 2. *Fix integers $\Delta, r \geq 2$, $0 < \beta < 1$, and $\sim \in \{\sim_{\mathrm{add}}, \sim_{\mathrm{rep}}, \sim_{\mathrm{sw}}\}$. For every public $N \geq 1$ and $\varepsilon \geq 0$, there exists a randomized algorithm that is pure ε-DO under $\sim$ on $\mathcal{D}_{N,\Delta}$ and, for every $G \in \mathcal{D}_{N,\Delta}$ and public source $s \in V(G)$, outputs $\mathsf{B}_r(G, s)$ with probability at least $1 - \beta$. Its expected access costs satisfy*

$$\mathbb{E}[T_{\mathrm{read}}(G, s)] \leq \mathbb{E}[T_{\mathrm{total}}(G, s)] = \mathrm{O}_{\Delta,r,\beta}\left(1 + N \exp\left(-\frac{\varepsilon}{C^{\sim}_{\Delta,r}}\right)\right). \tag{3}$$

Proof. We analyze Algorithm 3.

Correctness. Let A_i and S_i be the discovered and sampled sets in round i, and let A_r be the set of genuine labels in the final buffer. The algorithm implements the update

$$A_{i+1} = A_i \cup \bigcup_{v \in A_i \cap S_i} N_G(v).$$

Since $A_1 = \mathsf{B}_1(G, s)$, this gives $A_i \subseteq \mathsf{B}_i(G, s)$ and $|A_i| \leq M_i$ for every round. No insertion overflows the buffer, since all intermediate buffer contents in round i are subsets of A_{i+1}, whose size is at most $M_{i+1} \leq M_r$.

If all previous rounds sampled their entire discovered sets, then $A_i = \mathsf{B}_i(G, s)$. Conditional on this history, the probability of sampling all of A_i is $p^{|\mathsf{B}_i(G,s)|}$, and doing so gives $A_{i+1} = \mathsf{B}_{i+1}(G, s)$. Multiplying these conditional probabilities yields

$$\mathbb{P}[A_r = \mathsf{B}_r(G, s)] \geq p^{\sum_{i=1}^{r-1} |\mathsf{B}_i(G,s)|} \geq p^{\sum_{i=1}^{r-1} M_i} = 1 - \beta.$$

Sensitivity. Fix neighboring graphs $G \sim G'$ and a common sampling history $S_1, \ldots, S_{i-1}$. Let A_i, A'_i be the discovered sets. We show by induction that

$$\max\{|A_i \setminus A'_i|, |A'_i \setminus A_i|\} \leq \sigma_i. \tag{4}$$

For $i = 1$, each neighboring relation changes the source neighborhood by at most one vertex in either direction, so the bound holds with $\sigma_1 = 1$.

For the induction step, both executions use the same next sample S_i. At most σ_i vertices are present in A_i but not in A'_i. Each such vertex can account for itself and, if sampled, at most Δ newly reached neighbors, giving a total contribution of at most $(\Delta+1)\sigma_i$ vertices to $A_{i+1} \setminus A'_{i+1}$. Now consider a vertex in $A_{i+1} \setminus A'_{i+1}$ not accounted for above. It must have been discovered from a sampled vertex in $A_i \cap A'_i$. The edge used for this discovery cannot belong to G', since otherwise the vertex would also belong to A'_{i+1}.

Each such edge contributes at most one new difference vertex, since an edge with two common sampled endpoints has both endpoints in the discovered sets already. There is at most one edge in $E(G) \setminus E(G')$ for add/delete or replacement, and at most two for a 2-switch, adding at most κ vertices. The same argument bounds the reverse difference. Together with the size bound M_{i+1} on both next-round sets, this proves (4) by the definition of σ_{i+1}.

Privacy. For fixed public parameters, the complete access trace is determined by $\boldsymbol{S} = (S_1, \ldots, S_{r-1})$: initialization, counting, and final output use fixed scans, while sampled labels are processed in increasing order with prescribed record and buffer scans independent of membership bits and insertion outcomes.

Conditional on a common history, the round-i samples have distributions μ_{A_i} and $\mu_{A'_i}$. Lemma 3 and (4) therefore give

$$\frac{\mathbb{P}_G[\boldsymbol{S}]}{\mathbb{P}_{G'}[\boldsymbol{S}]} = \prod_{i=1}^{r-1} \frac{\mu_{A_i}(S_i)}{\mu_{A'_i}(S_i)} \leq \exp\left(\eta \sum_{i=1}^{r-1} \sigma_i\right) = e^{\varepsilon}.$$

This bound holds for every history and in both directions, proving pure ε-DP for $\boldsymbol{S}$ and hence pure ε-DO for its induced access trace.

Access cost and storage. The pathwise costs satisfy

$$T_{\text{read}} = \Delta\left(1 + \sum_{i=1}^{r-1} |S_i|\right),$$
$$T_{\text{total}} = \mathrm{O}\left(rM_r + \Delta M_r\left(1 + \sum_{i=1}^{r-1} |S_i|\right)\right). \tag{5}$$

For every round, conditional on the preceding history,

$$\mathbb{E}[|S_i| \mid A_i] = p|A_i| + q(N - |A_i|) \le M_i + Nq.$$

Summing over the rounds and using $q \le \frac{p}{1-p}e^{-\eta}$, we obtain

$$\mathbb{E}\left[\sum_{i=1}^{r-1} |S_i|\right] \le \sum_{i=1}^{r-1} M_i + (r-1)N\frac{p}{1-p}\exp\left(-\frac{\varepsilon}{C^{\sim}_{\Delta,r}}\right).$$

For fixed Δ, r, β, the quantities M_i and $\frac{p}{1-p}$ are constants. Substituting this bound into (5) leads to (3).

The buffer occupies M_r workspace cells, and the parameter recurrence, iterator, and scans need only a constant number of private registers. Public-label enumeration and coin generation use register-only computation. The bound concerns data-memory accesses, not the total instruction count. □

To illustrate how the radius r affects the access bound, we evaluate the sensitivity factor $C^{\sim}_{\Delta,r}$. For $r = 2$, we have $C^{\sim}_{\Delta,2} = 1$, which indicates that the upper bound coincides with the lower bound in Theorem 1. In other words, for all three neighboring relations and valid ϵ, δ, β, the optimal 2-hop query incurs $\Theta_{\Delta,\beta}(1 + Ne^{-\varepsilon})$ cell accesses.

For $r \ge 3$ the bound is not tight anymore. Specifically, for $r = 3$, we have

$$C^{\sim_{\text{add}}}_{\Delta,3} = C^{\sim_{\text{rep}}}_{\Delta,3} = 1 + \min\{1 + \Delta^2, \Delta + 2\} \le \Delta + 3,$$
$$C^{\sim_{\text{sw}}}_{\Delta,3} = 1 + \min\{1 + \Delta^2, \Delta + 3\} \le \Delta + 4.$$

Thus, for $r = 3$, the upper bound under all three neighboring relations is

$$\mathbb{E}[T_{\text{total}}] = \mathrm{O}_{\Delta,\beta}\left(1 + Ne^{-\varepsilon/(\Delta+4)}\right).$$

More generally, for fixed $\Delta, r \ge 2$ and $0 < \beta < 1/2$, the sampling upper bound is $o(N)$ whenever $\varepsilon \to \infty$. For $\varepsilon = \mathrm{O}(1)$ and $0 \le \delta \le (1 - 2\beta)/2$, Corollary 1 and the fully oblivious baseline of Proposition 2 give an optimal expected cost of $\Theta(N)$ for both input reads and total accesses, so constant privacy budgets offer no asymptotic improvement over full obliviousness.

Zero-error tradeoff. Finally, we discuss the affect of error bound β. The previous discussion on ϵ and r shows that larger ϵ may achieve $o(N)$ cost when $\beta > 0$. With zero error, however, the optimal expected access cost remains linear in N whenever δ is bounded away from 1, regardless of how ε grows with N. In particular, this holds under pure DO. The following corollary gives the precise dependence on δ.

Corollary 2 (Zero-error curve). *Fix $\Delta, r \geq 2$, $\varepsilon \geq 0$, and $0 \leq \delta \leq 1$. Consider add/delete privacy on $\mathcal{D}_{N,\Delta+1}$ or degree-preserving 2-switch privacy on $\mathcal{D}_{N,\Delta}$. For sufficiently large N, every $H \in \mathcal{D}_{N,\Delta}$, and every source s, the optimal pointwise expected zero-error cost is*

$$\Theta_{\Delta,r}\big(1 + N(1-\delta)\big).$$

The optimum is taken separately for T_{read} and T_{total} over algorithms that are correct with probability one and (ε, δ)-DO throughout the selected domain.

Proof. Taking $\beta = 0$ in Theorem 1 and using $\Lambda(\varepsilon, \delta, 0) = 1 - \delta$ gives the $\Omega_{\Delta,r}(N(1-\delta))$ term. For sufficiently large N, the ball around s contains a vertex other than s and omits another vertex. Exchanging these two labels gives a graph in the same domain with a different answer. Coupled executions on the two graphs are identical until the first input read. Zero error therefore forces an input read with probability one, giving the additive constant.

For the upper bound, the algorithm chooses the fully oblivious construction of Proposition 2 with probability $1 - \delta$, and ordinary exact truncated BFS with probability δ. Both use the degree cap of the selected domain. The BFS branch stores its discovered set in observable workspace and, for fixed Δ, r, uses $\mathrm{O}_{\Delta,r}(1)$ total accesses and a constant number of private registers. If Q is the graph-independent trace law of the first branch and R_G is that of the BFS branch, the complete trace has law

$$\mu_G = (1-\delta)Q + \delta R_G.$$

Thus $d_{\mathrm{TV}}(\mu_G, \mu_{G'}) \leq \delta$ for every pair of inputs with the same public parameters. Both branches are exact, and their mixture is $(0, \delta)$-DO with expected read and total costs $\mathrm{O}_{\Delta,r}(1 + N(1-\delta))$. □

6 Differentially Oblivious Preprocessing

The preceding sections study a single query from the raw graph. For repeated queries on a graph that does not change, we can build its neighborhoods once and reuse them. This section constructs an index before any query source is known, with one fixed-length row per source. A query reads its source's entire row and privately emits the genuine labels. The index is stored in workspace, with private contents and observable addresses.

Similar to the previous sections, our construction begins with the padded adjacency records and auxiliary state independent of the graph. We denote the total preprocessing cost by P and the cost of each subsequent query by Q.

Algorithm 4: BUCKETINDEX$(G, r, p, \varepsilon, \sim)$

Input: Graph $G \in \mathcal{D}_{N,\Delta}$ with $\Delta \geq 2$ in padded adjacency records; public radius $r \geq 2$, sampling probability $p \in (0,1)$, privacy budget $\varepsilon \geq 0$, and neighboring relation $\sim \in \{\sim_{\mathrm{add}}, \sim_{\mathrm{rep}}, \sim_{\mathrm{sw}}\}$.

Output: An $N \times M_r$ index $\mathcal{I}$ stored in workspace.

Initialize the parameters:

$$a_\sim \leftarrow \begin{cases} 2, & \sim = \sim_{\mathrm{sw}}, \\ 1, & \sim \in \{\sim_{\mathrm{add}}, \sim_{\mathrm{rep}}\}, \end{cases} \qquad L^{\sim}_{\Delta,r} \leftarrow 2a_\sim \sum_{t=0}^{r-2} (t+1)(r-1-t)(\Delta-1)^t,$$

$$\eta \leftarrow \frac{\varepsilon}{L^{\sim}_{\Delta,r}}, \quad q \leftarrow \frac{p}{p + e^{\eta}(1-p)}, \quad b \leftarrow \min\{N, \lfloor 1/q \rfloor\};$$

Partition labels into public intervals $V_1, \ldots, V_b$ of sizes differing by at most 1;
$\mathsf{bucket}(v) \leftarrow j$ for $v \in V_j$;
Initialize $A_1(s) \leftarrow \{s\} \cup N_G(s)$ for all $s \in V$;
for $i \leftarrow 1$ **to** $r-1$ **do**
  $A_{i+1}(s) \leftarrow A_i(s)$ for all $s \in V$;
  for $s \leftarrow 1$ **to** N **do**
    $D_i(s) \leftarrow \{\mathsf{bucket}(v) : v \in A_i(s)\}$;
    Form $J_i(s)$ by including each $j \in [b]$ independently with probability
    $\begin{cases} p, & j \in D_i(s), \\ q, & j \notin D_i(s); \end{cases}$
  for $j \leftarrow 1$ **to** b **do**
    Update all sources s with $j \in J_i(s)$ in one batch:

$$A_{i+1}(s) \leftarrow A_{i+1}(s) \cup \bigcup_{v \in A_i(s) \cap V_j} N_G(v).$$

return *the index with rows* $\mathcal{I}[s] = A_r(s)$ *for all* $s \in V$;

Algorithm 4, BUCKETINDEX, expands neighborhoods for all sources together. For each source, it samples buckets containing discovered vertices with probability p and other buckets with probability $q \leq p$. Requests from different sources are grouped by bucket and processed together, sharing the work of expansion and keeping each sorting batch small. The input p controls accuracy, while q is chosen using the privacy budget. We write $A_i(s)$ for the vertices discovered from source s at the start of round i.

Intuitively, in Algorithm 4, once all bucket selections are fixed, every memory address and read/write operation is fixed as well. Thus privacy can be proved by analyzing the distribution of the selections. To implement the algorithm with this property, we scan the input once, then build each M_r-cell row from the workspace copy by fixed scans, placing genuine labels before dummy padding. Each row represents the set of its genuine labels. Each round copies the current table to a separate table, applies all updates there, and then swaps their roles.

We describe how the sampling and batch updates preserve this property, with all temporary arrays kept in observable workspace.

Generating and grouping the selected pairs. For each source s, we form $D_i(s)$ in an M_r-cell padded array using $\mathrm{O}(M_r^2)$ accesses in a fixed pattern. Lemma 2 then generates $J_i(s)$ with fresh randomness in $\mathrm{O}(M_r(|J_i(s)|+1))$ accesses. Let $X_{i,j} = |\{s : j \in J_i(s)\}|$ and $Z_i = \sum_{j=1}^{b} X_{i,j}$. We group the selected pairs (s, j) by bucket in $\mathrm{O}(b + Z_i)$ time, with the access pattern determined by these pairs.

Processing a bucket. For each selected pair (s, j), a full scan of the M_r-cell row $A_i(s)$ produces one request per slot, keyed by v if the slot contains $v \in V_j$ and by the dummy key 0 otherwise. Thus each pair generates exactly M_r requests.

For each port $k \in \{1, 2, \ldots, \Delta\}$, the requests are joined with the table

$$\{(u, \mathsf{Record}[u][k]) : u \in V_j\} \cup \{(0, \perp)\},$$

read from the workspace copy of the graph. An oblivious sort orders table and request records by key, placing a table record before requests with the same key. A complete read–write scan then retains the current table value in a register and attaches it to the requests. A second oblivious sort places the $M_r X_{i,j}$ replies first in their original order, using the pair and row-slot positions stored with each request.

Each reply is inserted through a full scan and rewrite of the next-round row for the source specified by its pair. The row changes only if the returned label is genuine and absent; dummy replies cause the same scans and writes. Restoring request order before these updates ensures that the sequence of source-row accesses depends only on the selected pairs. The same procedure handles each port in turn.

Let $\ell := \lceil N/b \rceil$ be the maximum bucket size. Each sort acts on $m := |V_j| + 1 + M_r X_{i,j}$ records and uses a deterministic sorting network of size $\mathrm{O}(m \log(2+m))$ [12]. Every comparator reads and writes a fixed number of cells. The bucket therefore costs

$$\mathrm{O}_{\Delta,r}((\ell + X_{i,j}) \log(2 + \ell + X_{i,j})) \tag{6}$$

accesses. All scans and masked insertions have fixed lengths, whether or not a request is genuine.

For fixed public parameters, the complete preprocessing trace is consequently a deterministic function of

$$\boldsymbol{J} := (J_i(s) : 1 \le i < r,\ s \in \{1, 2, \ldots, N\}). \tag{7}$$

The sampled pairs determine routing addresses and bucket lengths, while oblivious sorting and fixed scans hide the private keys, matching results, and row contents. As in Section 5, the cost counts data-memory accesses. Enumerating public labels and generating biased coins can take additional register-only computation.

6.1 Analysis

The privacy analysis compares all discovered sets under a common sampling history. Each differing source–vertex pair has a short witness path through one of at most $a_\sim$ deleted edges in either direction of comparison. Bounded degree and radius therefore limit the number of such pairs across all sources, giving a sensitivity factor $L^{\sim}_{\Delta,r}$ independent of N.

Lemma 4. *Consider two executions of Algorithm 4 on neighboring graphs G, G' with the same bucket selections in the first $i-1$ rounds, where $1 \le i < r$. Superscripts in $A_i^G(s)$ and $D_i^G(s)$ indicate the graph used in the execution. The discovered sets satisfy*

$$\sum_{s\in V} \left| A_i^G(s) \setminus A_i^{G'}(s) \right| \le d_i = 2a_\sim \sum_{t=0}^{i-1} (t+1)(\Delta-1)^t. \tag{8}$$

The same bound holds with A_i replaced by D_i and with G, G' interchanged.

Proof. Fix s and $v \in A_i^G(s) \setminus A_i^{G'}(s)$. Tracing predecessors at first discovery gives a simple witness path from s to v of length at most i. Each edge was traversed at a permitted round, except for the unconditional first hop. If all its edges also belonged to G', the same bucket selections would discover v there as well: an earlier discovery cannot prevent a later expansion, because discovered vertices are retained. Thus every pair counted on the left of (8) has a witness path containing an edge of $E(G) \setminus E(G')$.

For a fixed orientation $u \to v$ of such an edge, a simple witness path has a prefix of a edges and a suffix of c edges, where $a + c \le i - 1$. The prefix, read backward from u, cannot first use $\{u, v\}$; likewise the suffix cannot immediately return to u. The numbers of possible prefixes and suffixes are therefore at most $(\Delta-1)^a$ and $(\Delta-1)^c$. Hence this oriented edge lies on witness paths for at most

$$\sum_{a+c\le i-1} (\Delta-1)^{a+c} = \sum_{t=0}^{i-1} (t+1)(\Delta-1)^t$$

ordered source–vertex pairs. There are at most $a_\sim$ deleted edges and two orientations per edge, giving (8). Exchanging G, G' gives the reverse bound. Finally, each bucket in $D_i^G(s) \setminus D_i^{G'}(s)$ contains a distinct vertex in $A_i^G(s) \setminus A_i^{G'}(s)$, so the same bound applies to the bucket sets. □

To state the accuracy guarantee, let $H_{\Delta,r} := \sum_{i=1}^{r-1} M_i$, where M_i is the ball-size bound from Section 2.

Theorem 3. *For fixed $\Delta, r \ge 2$, public $p \in (0,1)$, and $\varepsilon \ge 0$, Algorithm 4 is pure ε-DO on $\mathcal{D}_{N,\Delta}$ under $\sim$. It constructs an NM_r-cell index with*

$$\mathbb{E}[P] = \mathrm{O}_{\Delta,r}(N \log(2 + Nq)) \quad \text{and} \quad Q = M_r. \tag{9}$$

The index is correct for each fixed source with probability at least $p^{H_{\Delta,r}}$, and for all sources simultaneously with probability at least $p^{NH_{\Delta,r}}$.

Proof. Correctness. The batched lookups give the update

$$A_{i+1}(s) = A_i(s) \cup \bigcup_{\substack{v \in A_i(s) \\ \mathsf{bucket}(v) \in J_i(s)}} N_G(v).$$

Since the current rows stay unchanged throughout each round and $A_1(s) = \mathsf{B}_1(G, s)$, induction gives $A_i(s) \subseteq \mathsf{B}_i(G, s)$ and $|A_i(s)| \le M_i$ on every run. All intermediate next-round rows are subsets of $A_{i+1}(s)$, so masked insertion cannot overflow. For a fixed source whose previous rounds succeeded, $A_i(s) = \mathsf{B}_i(G, s)$. Selecting every bucket in $D_i(s)$ expands this entire set and produces $\mathsf{B}_{i+1}(G, s)$. Conditional on the preceding history, the probability of this event is $p^{|D_i(s)|} \ge p^{M_i}$. Multiplying the conditional bounds gives $p^{H_{\Delta,r}}$ for that source. Requiring the event for all sources in every round instead gives $p^{NH_{\Delta,r}}$, establishing both accuracy guarantees.

Privacy. We compare the two executions after any common history before round i. Let $\mathcal{D}_i(G) := \{(s, j) : j \in D_i^G(s)\}$. Conditional on that history, the selected-pair set $\{(s, j) : j \in J_i(s)\}$ is a product sample on $\{1, 2, \ldots, N\} \times [b]$, with probability p on $\mathcal{D}_i(G)$ and q elsewhere. Lemma 3, applied to this universe, bounds the likelihood ratio in terms of the difference between the active sets. By Lemma 4, that difference is at most d_i in either direction. The conditional likelihood ratio for any round-i selection is therefore at most $e^{\eta d_i}$. Multiplying over the rounds gives

$$\frac{\mathbb{P}_G[\boldsymbol{J}]}{\mathbb{P}_{G'}[\boldsymbol{J}]} \le \exp\left(\eta \sum_{i=1}^{r-1} d_i\right) = \exp\bigl(\eta L^{\sim}_{\Delta,r}\bigr) = e^{\varepsilon}.$$

All histories have positive probability, and exchanging the graphs gives the reverse bound. Since the complete access trace is a function of (7), this proves pure ε-DO. Each subsequent query appends the fixed scan of its public source's row. For sources fixed in advance or chosen using only the public transcript and graph-independent randomness, these scans are post-processing of the protected trace and preserve the same guarantee.

Expected access cost. All the following expectations are conditional on the preceding rounds. The choice of b in Algorithm 4 gives $bq \le 1$, $b\ell \le 2N$, and $\ell \le 2Nq + 1$. Thus each source selects at most 1 bucket containing no discovered vertex in expectation, and each bucket has $\mathrm{O}(1 + Nq)$ vertices. Since $|D_i(s)| \le M_i$, we have $\mathbb{E}[Z_i] \le \sum_{s=1}^{N} \bigl(|D_i(s)| + bq\bigr) \le N(M_i + 1)$. Thus generating and grouping the selected pairs costs $\mathrm{O}_{\Delta,r}(N)$ in expectation.

For a fixed bucket V_j, at most $M_i|V_j| \le M_i\ell$ sources have $j \in D_i(s)$. To see this, such a source has some $v \in V_j \cap A_i(s) \subseteq V_j \cap \mathsf{B}_i(G, s)$. Undirectedness implies $s \in \mathsf{B}_i(G, v)$, and every such ball has at most M_i vertices. Consequently, $X_{i,j}$ is a sum of independent Bernoulli variables with mean

$$\lambda_j := \mathbb{E}[X_{i,j}] \le M_i\ell + Nq.$$

The sorting cost also depends on the variation in this load. For a sum $X = \sum_s B_s$ of independent Bernoulli variables with mean λ, writing $X_{-s} = X - B_s$ allows

us to use independence and concavity of the logarithm:

$$\begin{aligned}\mathbb{E}[X\log(2+\ell+X)] &= \sum_s \mathbb{P}[B_s=1]\,\mathbb{E}[\log(3+\ell+X_{-s})]\\ &\le \lambda\log(3+\ell+\lambda),\\ \ell\mathbb{E}[\log(2+\ell+X)] &\le \ell\log(2+\ell+\lambda).\end{aligned}$$

Applying these inequalities to (6) bounds the expected bucket cost by

$$\mathrm{O}_{\Delta,r}((\ell+Nq)\log(2+\ell+Nq))\,.$$

Summing over the buckets yields $\mathrm{O}_{\Delta,r}(N\log(2+Nq))$ expected accesses per round. Initialization and table copying cost $\mathrm{O}_{\Delta,r}(N)$; there are $r-1$ rounds. This proves (9).

Storage and queries. The index contains NM_r cells and each query reads one complete row. The current and next-round tables, selected-pair lists, request buffers, and any sorting-control state all reside in observable workspace. Their accesses have been included above. Only a fixed number of labels, counters, record fields, and flags are retained privately at any time. A record for a batched lookup has a constant number of words; ports are processed sequentially rather than retaining an entire adjacency record in private registers. □

The following choices of p translate these success probabilities into an error bound β.

Corollary 3 (Accuracy and preprocessing cost). *For $r \ge 2$, $0 < \beta < 1$, and $L = L^{\sim}_{\Delta,r}$, the construction has the following two parameterizations, both with $Q = M_r$ and pure ε-DO.*

1. *With $p = (1-\beta)^{1/H_{\Delta,r}}$, every fixed source has error probability at most β, and we have*
$$\mathbb{E}[P] = \mathrm{O}_{\Delta,r,\beta}\Big(N\log(2+Ne^{-\varepsilon/L})\Big)\,.$$
In particular, $\varepsilon \ge L\log N$ gives $\mathbb{E}[P] = \mathrm{O}_{\Delta,r,\beta}(N)$.
2. *With $p = (1-\beta)^{1/(NH_{\Delta,r})}$, the entire index is correct with probability at least $1-\beta$, and we have*
$$\mathbb{E}[P] = \mathrm{O}_{\Delta,r,\beta}\Big(N\log\Big(2+\min\{N, N^2e^{-\varepsilon/L}\}\Big)\Big)\,.$$
In particular, $\varepsilon \ge 2L\log N$ gives $\mathbb{E}[P] = \mathrm{O}_{\Delta,r,\beta}(N)$.

Proof. The accuracy statements follow from Theorem 3. For every integer $h \ge 1$, we have $1-(1-\beta)^{1/h} \ge \beta/h$. Therefore, when $p = (1-\beta)^{1/h}$, we have $q \le \min\{1, (h/\beta)e^{-\varepsilon/L}\}$. Substituting $h = H_{\Delta,r}$ or $h = NH_{\Delta,r}$ into (9) gives the two cost bounds. At the respective stated privacy budgets, $Nq \le H_{\Delta,r}/\beta$, proving the linear bounds. □

These probabilities refer to the single construction of the index. Repeated queries to a source read the same stored row and return the same answer. Under the simultaneous guarantee, one event of probability at least $1-\beta$ ensures that every subsequent query is correct.

For $T \geq 1$ queries, the expected total cost is $\mathbb{E}[P]+TM_r$, giving an amortized cost of $\mathbb{E}[P]/T + M_r$ per query.

Size of the sensitivity factor. The factor $L^{\sim}_{\Delta,r}$ determines the sufficient privacy budgets in Corollary 3. For the first three nontrivial radii,

$$\begin{aligned} L^{\sim\mathrm{add}}_{\Delta,2} &= L^{\sim\mathrm{rep}}_{\Delta,2} = 2, \\ L^{\sim\mathrm{add}}_{\Delta,3} &= L^{\sim\mathrm{rep}}_{\Delta,3} = 4\Delta, \\ L^{\sim\mathrm{add}}_{\Delta,4} &= L^{\sim\mathrm{rep}}_{\Delta,4} = 6\Delta^2 - 4\Delta + 4. \end{aligned}$$

The 2-switch factors are twice these values. More generally, we have

$$L^{\sim}_{2,r} = \frac{a_{\sim}}{3}(r^3 - r), \quad L^{\sim}_{\Delta,r} = \Theta\big(a_{\sim} r(\Delta-1)^{r-2}\big) \quad (\Delta \geq 3).$$

For fixed degree and radius, these factors are constants. The resulting sufficient budgets are $L^{\sim}_{\Delta,r}\log N$ for correctness at each fixed source and $2L^{\sim}_{\Delta,r}\log N$ for simultaneous correctness. These coefficients need not be optimal.

Comparison with fully oblivious preprocessing. The same batched propagation with a single bucket, selected deterministically for every source in every round, gives an exact fully oblivious $\mathrm{O}_{\Delta,r}(N\log N)$ construction. The sampled construction improves on this bound at the stated privacy budgets by allowing a small probability of error. We do not prove an $\Omega(N\log N)$ lower bound for fully oblivious preprocessing.

7 Conclusion and Future Directions

We establish access bounds for graph ball queries from raw adjacency records, with all preprocessing accesses charged. For fixed $\Delta, r \geq 2$ and $0 < \beta < 1/2$, we prove a pointwise $\Omega_{\Delta,r,\beta}(1 + Ne^{-\varepsilon})$ expected input-read lower bound on every sufficiently large graph in $\mathcal{D}_{N,\Delta}$ and every public source, provided $0 \leq \delta \leq (1-2\beta)/2$. This holds under all three neighboring relations. Pure ε-DO sampling matches the bound on total observable accesses for $r = 2$.

For repeated queries, our pure ε-DO preprocessing constructs a linear-size index with constant query cost for fixed Δ, r. All rows are correct simultaneously with probability at least $1-\beta$. For fixed β, the expected preprocessing cost is linear in N once ε exceeds a sufficiently large constant multiple of $\log N$.

Beyond our work, two main questions remain open: whether the $e^{-\varepsilon}$ dependence can be attained for every fixed $r \geq 3$, and what the optimal tradeoff is between protected preprocessing cost and subsequent query cost.

References

1. Arasu, A., Kaushik, R.: Oblivious query processing. In: Proceedings of the 17th International Conference on Database Theory (ICDT 2014). pp. 26–37 (2014)
2. Bayati, M., Gamarnik, D., Katz, D., Nair, C., Tetali, P.: Simple deterministic approximation algorithms for counting matchings. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC). pp. 122–127 (2007)
3. Beimel, A., Nissim, K., Zaheri, M.: Exploring differential obliviousness. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 145, pp. 65:1–65:20 (2019)
4. Chan, T.H.H., Chung, K.M., Maggs, B.M., Shi, E.: Foundations of differentially oblivious algorithms. In: Chan, T.M. (ed.) Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019). pp. 2448–2467 (2019)
5. Chu, S., Zhuo, D., Shi, E., Chan, T.H.H.: Differentially oblivious database joins: Overcoming the worst-case curse of fully oblivious algorithms. In: Tessaro, S. (ed.) 2nd Conference on Information-Theoretic Cryptography (ITC 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 199, pp. 19:1–19:24 (2021)
6. Dwork, C.: Differential privacy. In: International colloquium on automata, languages, and programming. pp. 1–12 (2006)
7. Erling, O., Averbuch, A., Larriba-Pey, J., Chafi, H., Gubichev, A., Prat, A., Pham, M.D., Boncz, P.: The LDBC social network benchmark: Interactive workload. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. pp. 619–630 (2015)
8. Fan, X., Chen, K., Yu, J., Zhu, X., Chen, Y., Zhang, H., Xu, W.: GORAM: Graph-oriented ORAM for efficient ego-centric queries on federated graphs. Proceedings of the VLDB Endowment **18**(10), 3601–3614 (2025)
9. Feng, W., Cao, X., O'Neill, A., Yang, C.: Enabling index-free adjacency in oblivious graph processing with delayed duplications. Proceedings of the VLDB Endowment **19**(8), 1826–1839 (2026)
10. Ghareh Chamani, J., Demertzis, I., Papadopoulos, D., Papamanthou, C., Jalili, R.: GraphOS: Towards oblivious graph processing. Proceedings of the VLDB Endowment **16**(13), 4324–4338 (2023)
11. Goldreich, O.: Towards a theory of software protection and simulation by oblivious rams. In: Proceedings of STOC. pp. 182–194 (1987)
12. Goodrich, M.T.: Zig-zag sort: A simple deterministic data-oblivious sorting algorithm running in $O(n \log n)$ time. In: Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing (2014)
13. Islam, M.S., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In: Proceedings of NDSS (2012)
14. Mazloom, S., Gordon, S.D.: Secure computation with differentially private access patterns. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 490–507 (2018)
15. Patel, S., Persiano, G., Yeo, K.: What storage access privacy is achievable with small overhead? In: Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 2019). pp. 182–199 (2019)
16. Persiano, G., Yeo, K.: Limits of preprocessing for single-server PIR. In: Naor, J., Buchbinder, N. (eds.) Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2022). pp. 2522–2548 (2022)

17. Persiano, G., Yeo, K.: Lower bound framework for differentially private and oblivious data structures. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology–EUROCRYPT 2023, Part I. Lecture Notes in Computer Science, vol. 14004, pp. 487–517 (2023)
18. Sahu, S., Mhedhbi, A., Salihoglu, S., Lin, J., Özsu, M.T.: The ubiquity of large graphs and surprising challenges of graph processing: Extended survey. The VLDB Journal **29**(2–3), 595–618 (2020)
19. Weitz, D.: Counting independent sets up to the tree threshold. In: Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC). pp. 140–149 (2006)
20. Wu, Z., Dong, W., Hu, X.: Differentially oblivious multi-way join. Proceedings of the ACM on Management of Data **4**(3), 1–26 (2026)
21. Yeo, K.: Lower bounds for (batch) PIR with private preprocessing. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology–EUROCRYPT 2023, Part I. Lecture Notes in Computer Science, vol. 14004, pp. 518–550 (2023)
22. Zhou, M., Shi, E., Chan, T.H.H., Maimon, S.: A theory of composition for differential obliviousness. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology–EUROCRYPT 2023, Part III. Lecture Notes in Computer Science, vol. 14006, pp. 3–34 (2023)

AI Disclosure

The main results of this work were developed primarily by the authors. We use ChatGPT to validate the results and draft part of the proofs. Besides, GPT was also used for polishing the texts. The authors take full responsibility for the content of the manuscript.