

EFFICIENT ALGORITHMS FOR ADVERSARIALLY ROBUST APPROXIMATE NEAREST NEIGHBOR SEARCH

Anonymous authors

Paper under double-blind review

ABSTRACT

We study the Approximate Nearest Neighbor (ANN) problem under a powerful adaptive adversary that controls both the dataset and a sequence of Q queries.

Primarily, for the high-dimensional regime of $d = \omega(\sqrt{Q})$, we introduce a sequence of algorithms with progressively stronger guarantees. We first establish a novel connection between adaptive security and *fairness*, leveraging fair ANN search (Aumüller et al., 2022) to hide internal randomness from the adversary with information-theoretic guarantees. To achieve data-independent performance, we then reduce the search problem to a robust decision primitive, solved using a differentially private mechanism (Hassidim et al., 2022) on a Locality-Sensitive Hashing (LSH) data structure. This approach, however, faces an inherent \sqrt{n} query time barrier. To break this barrier, we propose a novel concentric-annuli LSH construction that synthesizes these fairness and differential privacy techniques. The analysis introduces a new method for robustly releasing timing information from the underlying algorithm instances and, as a corollary, also improves existing results for fair ANN.

In addition, for the low-dimensional regime $d = O(\sqrt{Q})$, we propose specialized algorithms that provide a strong “for-all” guarantee: correctness on *every* possible query with high probability. We introduce novel metric covering constructions that simplify and improve prior approaches for ANN in Hamming and ℓ_p spaces.

1 INTRODUCTION

Randomness is a crucial tool in algorithm design, enabling resource-efficient solutions by circumventing the worst-case scenarios that plague deterministic approaches (Motwani & Raghavan, 1996). The classical analysis of such algorithms assumes an **oblivious** setting, where data updates and queries are fixed beforehand. However, this assumption breaks down in the face of an **adaptive adversary**, who can issue queries based on the algorithm’s previous outputs. These outputs can leak information about the algorithm’s internal randomness, allowing an adversary to construct query sequences that maliciously break the algorithm’s performance guarantees (Hardt & Woodruff, 2013; Gribelyuk et al., 2024).

Significant progress has been made in designing adversarially robust algorithms for **estimation problems**, where the output is a single value (Lai & Bayraktar, 2020; Hassidim et al., 2022; Chakrabarti et al., 2021; Attias et al., 2024; Ben-Eliezer et al., 2022a; Woodruff & Zhou, 2022; Cherapanamjeri et al., 2023). A common defense involves sanitizing the output, for example, by rounding or adding noise, often borrowing techniques from differential privacy to ensure the output reveals little about the algorithm’s internal state (Hassidim et al., 2022; Attias et al., 2024; Beimel et al., 2022). However, these techniques do not directly apply to **search problems**. In a search problem, the algorithm must return a specific element from a given dataset. Outputting a raw data point can leak substantial information, and there is no obvious way to add noise or otherwise obscure the output without violating the problem’s core constraint of returning a valid dataset element.

Perhaps the most fundamental search problem is *Approximate Nearest Neighbor (ANN) Search*, which has numerous applications ranging from data compression and robotics to DNA sequencing and anomaly detection to Retrieval-Augmented Generation (SantaLucia et al., 1996; Kalantidis & Avrithis, 2014; Ichnowski & Alterovitz, 2015; Verstrepen & Goethals, 2014; Tagami, 2017; Bergman et al., 2020; Han et al., 2024; Kitaev et al., 2020). Given a dataset S of n points in a metric

space $(\mathcal{M}, \|\cdot\|)$ and a radius $r > 0$, let $B_S(q, r) := \{p \in S : \|p - q\| \leq r\}$. Given a query point $q \in \mathcal{M}$ and approximation parameter $c \geq 1$, the goal is to build a data structure which finds a point in $B_S(q, cr)$ if $B_S(q, r) \neq \emptyset$. If $B_S(q, cr) = \emptyset$, the algorithm is required to answer \perp . Apart from queries, the dataset S itself may also be *obliviously updated* via additions or deletions of points.

Achieving the desired trade-off of sublinear query time and near-linear space has largely been possible only through randomization. Indeed, one of the most prominent family of algorithms for ANN is based on *Locality-Sensitive Hashing (LSH)*, which has been the subject of a long and fruitful line of research in the oblivious setting (Gionis et al., 1999; Jafari et al., 2021; Andoni, 2009; Andoni et al., 2018; 2017b; 2016; Andoni & Indyk, 2017; Andoni et al., 2017a; Indyk & Motwani, 1998; Broder et al., 1998; Andoni & Beaglehole, 2022). ANN Algorithms that rely on LSH achieve query time complexity of $\tilde{O}(dn^\rho)^1$ and space complexity $\tilde{O}(n^{1+\rho})$, where d is the dimension of \mathcal{M} and $\rho = \rho(c) \in (0, 1)$ is a fixed constant depending on c and the LSH construction².

The vulnerability of these classical randomized structures was recently highlighted by Kapralov et al. (2024), who demonstrated an attack on standard LSH data structures. They showed that an adaptive adversary can use a polylogarithmic number of queries to learn enough about the internal hash functions to force the algorithm to fail. Inspired by their work, which relies on certain structural properties of the dataset (e.g., an “isolated” point), we consider a powerful adversarial model where the *adversary chooses both the dataset and the sequence of queries*. We study the following question:

Can search problems like ANN be solved efficiently in the face of adversarial queries?

1.1 OUR RESULTS AND TECHNIQUES

We propose adversarially robust algorithms answering the above questions in two regimes:

1.1.1 $d = \omega(\sqrt{Q})$

When the metric space dimension is very large in the sense that $d = \omega(\sqrt{Q})$, we tackle the search problem by employing a suite of different strategies.

Robustness and Fairness We first recognize a connection between robustness and fairness. *Fair* ANN algorithms output a point uniformly at random from a set of valid near neighbor candidates. Such algorithms have already been rigorously studied in the context of LSH by Aumüller et al. (2022), who also studied notions of approximate fairness. We show that the robust ANN problem can be solved simply by invoking an algorithm for the exact fair ANN problem.

Theorem 1.1. *Let $n(q, r) := |B_S(q, r)|$ be the S -density of the r -ball centered at $q \in \mathcal{M}$. There exists an adversarially robust (c, r) -ANN algorithm that uses $O(n^{1+\rho(c)} \log^2(n) \log(Q))$ bits of space and $O(d \cdot (n^{\rho(c)} + \frac{n(q, cr)}{n(q, r)+1}) \log^2(n) \log(Q))$ time per query.*

Note that the space complexity of this algorithm does not scale with \sqrt{Q} , unlike our other approaches and also the algorithm of Feng et al. (2025). However, the query complexity depends on the density ratio D of points between the cr -ball and the r -ball for a query q . An adversary can craft a dataset where this ratio is large, severely degrading performance. This drawback is also shared by the algorithm of (Feng et al., 2025), though they exhibit a dependency on the density $s = n(q, cr)$, which is strictly greater than D (see Table 1).

Remark. *The link between fairness and robustness is not limited to ANN. From this perspective, fairness is not just a “nice to have” property, but is inextricably linked with security.*

Assumption-Free Searching via Bucketing To mitigate data dependencies, we propose a meta-algorithm that reduces a search problem to a weak decision problem. In this problem, positive instances correspond to the existence of r -close neighbors to a query q , while negative instances

¹We use the \tilde{O} notation to hide polylogarithmic factors.

²For example, when $\mathcal{M} = \{0, 1\}^k$ and $c \geq 1$ is the approximation parameter, the state-of-the-art construction of Andoni & Razenshteyn (2015) yields $\rho = \frac{1}{2c-1}$. We shall use ρ and $\rho(c)$ interchangeably.

showcase the absence of cr -close neighbors. Such a weak decision problem can be solved obliviously simply by using a classic LSH data structure \mathcal{D} . Unlike the search problem, an oblivious decider can be robustified by applying the well-known Differential Privacy (DP) obfuscation technique of (Hassidim et al., 2022): we maintain \sqrt{Q} copies of \mathcal{D} and combine their responses in a private manner with respect to the random bits of each copy.

To perform the search, we then partition $S \in \mathcal{M}^n$ into buckets of size roughly \sqrt{n} and instantiate a copy of the robust weak decider in each bucket. We can use these copies to identify a bucket that contains a suitable point to output and then exhaustively search that bucket to produce the final answer:

Theorem 1.2. *There exists an adversarially robust algorithm for the (c, r) -ANN problem, successfully answering up to Q queries with probability at least $1 - \Theta(\delta)$. The algorithm uses $\tilde{O}(n^{1+\rho/(2-\rho)}\sqrt{Q})$ space and $\tilde{O}(dn^{1/(2-\rho)})$ time per-query, where $\rho = \rho(c) \in (0, 1)$.*

Metric	Query Time	Space	Update Time
Theorem 1.1 (Fairness)	$\tilde{O}(d \cdot (D + n^\rho))$	$\tilde{O}(n^{1+\rho})$	$\tilde{O}(n^\rho)$
Theorem 1.2 (Bucketing)	$\tilde{O}(dn^{\frac{1}{2-\rho}})$	$\tilde{O}(\sqrt{Q} \cdot n^{\frac{1}{2-\rho}})$	$\tilde{O}(dn^{\frac{1-\rho}{2-\rho}}\sqrt{Q})$
Theorem 1.3 (Concentric Annuli) $\beta = \Theta(\frac{\log \log c}{\log c})$	$\tilde{O}(dn^\beta)$	$O(\sqrt{Q} \cdot n^{1+\beta})$	$\tilde{O}(dn^\beta\sqrt{Q})$
(Feng et al., 2025) ³	$O(d \cdot s \cdot n^\rho)$	$O(\sqrt{Q} \cdot s \cdot n^{1+\rho})$	$\tilde{O}(dn^\rho \cdot s \cdot \sqrt{Q})$

Table 1: Algorithms for (c, r) -ANN problem $\{0, 1\}^d$ under the Hamming distance, where $\rho = \frac{1}{2c-1}$

Breaking the \sqrt{n} Barrier via Concentric LSH Annuli Finally, the bucketing method yields a query time complexity that is always at least $O(\sqrt{n})$, which is not ideal considering that LSH methods can induce the exponent of n to be arbitrarily close to 0. To go beyond this barrier, we introduce a *concentric annuli construction*.

We partition the (r, cr) -annulus into several smaller, concentric sub-annuli and apply the fair ANN algorithm $\mathcal{A}_{\text{fair}}$ within each one. A simple counting argument guarantees that at least one of these sub-annuli must have a low point-density ratio, allowing $\mathcal{A}_{\text{fair}}$ to terminate efficiently. For each annulus that does not exceed the runtime threshold, we obtain an estimate to the probability that the corresponding $\mathcal{A}_{\text{fair}}$ copy terminates quickly. We call the termination times “timestamps”, and their analysis allows us to pick a favorable annulus to run a held-out *testing* copy of $\mathcal{A}_{\text{fair}}$. To maintain robustness however, we must also be careful not to release information regarding which annulus was used at each query. To do this we apply the DP robustification framework on the “timestamp” probabilities of each annulus-based, fair algorithm.

This algorithm is both assumption-free and enjoys a better runtime than $O(\sqrt{n})$. Our result holds for any metric space equipped with a family of LSH functions, though its runtime and space guarantees depend on the structure of that family.

Theorem 1.3. *There exists a robust algorithm for solving the (c, r) -ANN problem that uses space $\tilde{O}(\sqrt{Q} \cdot n^{1+\beta})$, where $\beta = \min_{k \in \mathbb{Z}_{\geq 1}} \max\{\rho(c^{1/k}), 1/k\}$. Any query takes $\tilde{O}(dn^\beta)$ time with probability at least 0.998.*

For many metric spaces, the value of β resolves nicely. For the hypercube $\{0, 1\}^d$ under the Hamming distance we have $\rho(c) = \frac{1}{2c-1}$, which yields $\beta = \Theta(\frac{\log c}{\log \log c}) \rightarrow 0$ as $c \rightarrow \infty$, which is not the case with the exponent $\frac{1}{2-\rho(c)}$ of Theorem 1.2. As a corollary, this technique also allows us to achieve purely sublinear time for a class of “relaxed” fair ANN problems.

³The work of Feng et al. (2025) concurrently studies the robust ANN problem. We present a comparison of our results with their algorithm, as well as a more extended discussion of related work, in Appendix A.

1.1.2 $d = O(\sqrt{Q})$: For-all ALGORITHMS

For low-dimensional metric spaces, we develop algorithms for ANN that provide a powerful *for-all guarantee*: with high probability, the data structure correctly answers *every possible* query $q \in \mathcal{M}$. Our approach builds on a discretization technique applied to an LSH data structure, a paradigm explored in prior work (Cherapanamjeri & Nelson, 2020; 2024; Bateni et al., 2024). We refine this line of research by introducing a novel, simpler metric covering construction, improving the space complexity by a logarithmic factor, and using sampling to improve the time complexity by a factor of d . We present our result for the Hamming hypercube below, including results for ℓ_p spaces in Appendix E.

Theorem 1.4. *For the (c, r) -ANN problem in the d -dimensional Hamming hypercube $\{0, 1\}^d$, there exists an algorithm that correctly answers all possible queries with at least 0.99 probability. The space complexity is $\tilde{O}(d \cdot n^{1+\rho+o(1)})$ and query time is $\tilde{O}(d \cdot n^\rho)$, where $\rho = \frac{1}{2c-1}$.*

Remark (The Price of For-All Algorithms). *Despite their remarkable guarantees, for-all algorithms have significant drawbacks. Their space complexity scales by a factor of d , making them intractable for high-dimensional metric spaces.*

2 PRELIMINARIES

The Adversarial Robustness Model An algorithm is adversarially robust if it correctly answers a sequence of adaptively chosen queries with high probability. This is formalized (Ben-Eliezer et al., 2022b) through the following interactive game:

Definition 2.1. *Consider the following game \mathcal{G} between an **Algorithm** (\mathcal{A}) and an **Adversary** (\mathcal{B}):*

1. **Setup Phase:** *The adversary chooses a dataset S . The algorithm \mathcal{A} then uses its private internal randomness $R_{\text{setup}} \in \{0, 1\}^*$ to preprocess S and build a data structure D . The adversary may know the code for \mathcal{A} but not the specific instance of R_{setup} .*
2. **Query Phase:** *The game proceeds for Q rounds. In each round $i \in [Q]$:*
 - *The adversary adaptively chooses a query q_i . This choice can depend on the dataset S and the history of all previous queries and their corresponding answers, $(q_1, a_1), \dots, (q_{i-1}, a_{i-1})$.*
 - *The algorithm \mathcal{A} uses its data structure D and potentially new private randomness $R_i \in \{0, 1\}^*$ to compute and return an answer a_i .*
3. **Winning Condition:** *The algorithm fails if there exists at least one round $i \in [Q]$ for which the answer a_i is an incorrect response to the query q_i .*

We say that an algorithm \mathcal{A} is δ -adversarially robust if for any dataset and any strategy the adversary can employ, the probability that the algorithm fails is at most δ . The probability is taken over the algorithm’s entire internal randomness $(R_{\text{setup}}, R_1, \dots, R_Q)$.

Approximate Nearest Neighbor Search and LSH In the *Nearest Neighbors* problem, we seek to find a point in our input dataset that minimizes the distance to some query point.

Definition 2.2 (ANN). *Let $c > 1$ and $r > 0$ be positive constants. In the (c, r) -Approximate Nearest-Neighbors Problem (ANN) we are given as input a set $S \subset M$ with $|S| = n$ and a sequence of queries $\{q_i\}_{i=1}^Q$ with $q_i \in \mathcal{M}$. For each query q_i , if there exists $p \in B_S(q_i, r)$, we are required to output some point $p' \in B_S(q_i, cr)$. If $B_S(q_i, cr) = \emptyset$, we are required to output \perp . In the case where $B_S(q_i, r) = \emptyset \neq B_S(q_i, cr)$ we can either output a point from $B_S(q_i, cr)$ or \perp . Our algorithm should successfully satisfy these requirements with probability at least $2/3$.*

A prevalent method for solving ANN is Locality Sensitive Hashing (LSH):

Definition 2.3 (Locality Sensitive Hashing, (Har-Peled et al., 2012)). *A hash family \mathcal{H} of functions mapping \mathcal{M} to a set of buckets is called a (c, r, p_1, p_2) -Locality Sensitive Hash Family (LSH) if the following two conditions are satisfied:*

- *If $x, y \in \mathcal{M}$ have $\|x - y\| \leq r$, then $\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \geq p_1$.*

- If $x, y \in \mathcal{M}$ have $\|x - y\| \geq cr$, then $\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq p_2$.

where $p_1 \gg p_2$ are parameters in $(0, 1)$. We often assume that computing h in a d -dimensional metric space requires $O(d)$ time. We assume that the LSH constructions we consider are **monotone**, which means that $\Pr[h(x) = h(y)]$ monotonically decreases as $\|x - y\|$ increases.

For instance, in the boolean hypercube $M = \{0, 1\}^d$ with $\|x - y\|$ being the number of positions $j \in [d]$ for which $x_j \neq y_j$, there is a simple monotone LSH family:

Lemma 2.4. Consider the family $\mathcal{H} = \{h_j\}_{j=1}^d$ consisting of functions that map $x \in \{0, 1\}^d$ to its j -th bit $x[j]$. Then, \mathcal{H} is a (c, r, p_1, p_2) -LSH where $p_1 = 1 - \frac{r}{d}$, and $p_2 = 1 - \frac{cr}{d}$.

Proof. $\Pr_{h \in \mathcal{H}}[h(x) = h(y)] = \Pr_{j \in [d]}[x[j] = y[j]] = \frac{d - \|x - y\|}{d} = 1 - \frac{\|x - y\|}{d}$. \square

Given a construction of a (c, r, p_1, p_2) -LSH for a metric space, we can solve the (c, r) -ANN problem by amplifying the LSH guarantees. This is done via an “OR of ANDs” construction: we sample $L := p_1^{-1} = n^\rho$ hash functions h_1, \dots, h_L for $\rho(c) \in (0, 1)$ by concatenating the outputs of $k = \lceil \log_{1/p_2} n \rceil$ “prototypical” LSH functions in \mathcal{H} , as shown in (Indyk & Motwani, 1998).

Theorem 2.5. If a d -dimensional metric space admits a (c, r, p_1, p_2) -LSH family, then we can solve the (c, r) -ANN problem on it using $O(n^{1+\rho})$ space and $O(dn^\rho)$ time per query, where $\rho = \frac{\log p_1}{\log p_2}$.

3 FAIRNESS IMPLIES ROBUSTNESS

We first establish a connection between robustness and fairness. We use the definition of fairness in ANN given by Aumüller et al. (2022) and argue that it is strong enough to guarantee robustness.

Definition 3.1. A data structure solves the **Exact Fair** (c, r) -ANN problem for a sequence of Q queries if, with probability at least $1 - \frac{1}{Qn}$ over $(R_{\text{setup}}, R_1, \dots, R_Q)$, it satisfies the following conditions for every query q_i in the sequence $i \in [Q]$:

1. **Fairness:** If $B_S(q_i, r) \neq \emptyset$, then the probability of returning any specific point $p' \in B_S(q_i, r)$ is exactly $\frac{1}{n(q_i, r)}$. If $B_S(q_i, r) = \emptyset$, the algorithm must answer \perp .
2. **Independence:** For each $i = 1, 2, \dots, Q$, conditioned on the algorithm’s success on queries q_1, \dots, q_{i-1} , the distribution of the answer for query q_i is statistically independent of the answers returned for all previous queries and of the setup randomness R_{setup} .

In their prior work, Aumüller et al. (2022) prove the following theorem:

Theorem 3.2. There exists an exact fair (c, r) -ANN data structure using $O(n^{1+\rho} \log^2(n) \log(Q))$ space and $O(d \cdot (n^\rho + \frac{n(q, cr)}{n(q, r) + 1}) \cdot \log^2(n) \log(Q))$ time per query in expectation.

Let $\mathcal{A}_{\text{fair}}$ be the exact fair ANN algorithm given by Theorem 3.2. We claim that this algorithm is also adversarially robust.

Claim 3.3 (Fairness Implies Robustness). $\mathcal{A}_{\text{fair}}$ is an $\frac{1}{n}$ -adversarially robust ANN algorithm.

Proof. WLOG we can treat the adversary \mathcal{B} of game \mathcal{G} (Definition 2.1) as deterministic by fixing its internal randomness to be the string that maximizes its probability of winning. Let $T_i = (q_1, a_1, \dots, q_i, a_i)$ be a transcript of the interaction between \mathcal{B} and $\mathcal{A}_{\text{fair}}$ until step i .

We claim that for each $i \in [Q]$, if we condition on T_i being such that the fair algorithm succeeds on all steps $t \leq i$, then $R_{\text{setup}} \perp\!\!\!\perp T_i$. This can easily be seen via induction: For $i = 0$, R_{setup} is trivially independent of the empty set and if we condition on success of transcript T_{i+1} we know via the induction hypothesis that $T_i \perp\!\!\!\perp R_{\text{setup}}$. Since the adversary is deterministic we get that $q_{i+1} \perp\!\!\!\perp R_{\text{setup}}$. Now, $a_{i+1} \sim \text{unif}(B_S(q_{i+1}, r))$ where the random coins in the uniform distribution are exactly the transcient randomness R_i , i.e. mutually independent from R_{setup} and q_{i+1} . Thus, a_{i+1} is also independent from R_{setup} and the claim follows.

Now, let F_i be the event of the fair algorithm failing in step i and S_i be the event of it succeeding on steps $1, \dots, i$. By the law of total probability, the probability of $\mathcal{A}_{\text{fair}}$ failing in the adaptive interaction

270 is:

$$271 \Pr[F_1] + \sum_{i=2}^Q \Pr[F_i | S_{i-1}] \cdot \Pr[S_{i-1}] \leq \frac{1}{nQ} + \sum_{i=2}^Q \Pr[F_i | S_{i-1}] \quad (1)$$

272 The event of failure in the i -th round depends only on R_{setup} and R_i . Conditioned on S_{i-1} we know
 273 that $T_{i-1} \perp \{R_{\text{setup}}, R_i\}$, so this situation is equivalent to failing on a single, obviously determined
 274 query, which happens with probability at most $\frac{1}{nQ}$. By Equation 1 the overall failure probability is
 275 at most $\frac{1}{n}$ for $\mathcal{A}_{\text{fair}}$. Since our argument was done for an adaptive adversary it is implied that $\mathcal{A}_{\text{fair}}$ is
 276 $\frac{1}{n}$ -adversarially robust. \square

277 4 ASSUMPTION-FREE ROBUST SEARCHING VIA BUCKETING

278 A major limitation of the fair algorithm is that it only works efficiently when the dataset does not
 279 induce a high density ratio, which is not guaranteed if S is picked by the adversary. Ideally, we aim
 280 to obtain sublinear algorithms that work without any assumptions on the input dataset. To do this,
 281 we introduce a search-to-decision framework:

282 4.1 WEAK DECISION ANN

283 **Definition 4.1 (WEAK-DECISION-ANN).** Consider the metric space \mathcal{M} and let $S \subseteq U$ with
 284 $|S| = n$ be an input point dataset. Let $r > 0, c > 1$ be two parameters and $q \in \mathcal{M}$ be an adaptively
 285 chosen query. If $B_S(q, r) \neq \emptyset$, then we should answer **1**. If $B_S(q, cr) = \emptyset$, we must answer **0**. In
 286 any other case, any answer is acceptable.

287 Let \mathcal{A} be an algorithm for solving the weak decision ANN problem, though not necessarily robustly.
 288 We can design an adversarially robust decider \mathcal{A}_{dec} by using \mathcal{A} , while only increasing the space by
 289 a factor of \sqrt{Q} . Adhering to the framework of Hassidim et al. (2022), we maintain $L = \tilde{\Theta}(\sqrt{Q})$
 290 copies of the data structures $\mathcal{D}_1, \dots, \mathcal{D}_L$ generated by \mathcal{A} using L independent random strings, and
 291 then for each query q we combine the answers of \mathcal{A} privately. As opposed to the original framework
 292 of (Hassidim et al., 2022), we do not need to use a private median algorithm, which simplifies the
 293 analysis. To keep the query time small, we utilize privacy amplification by subsampling (Theorem B.8).

301 **Algorithm 1** The robust decider \mathcal{A}_{dec} (based on an oblivious decider \mathcal{A})

- 302 1: **Inputs:** Random string $R = r_1 \circ r_2 \circ \dots \circ r_L$.
 - 303 2: **Parameters:** Number of queries Q , number of copies L , number of sampled indices k .
 - 304 3: Receive input dataset $S \subseteq U$ from the adversary, where $n = |S|$.
 - 305 4: Initialize $\mathcal{D}_1, \dots, \mathcal{D}_L$ where $\mathcal{D}_i \leftarrow \mathcal{A}(S)$ on random string r_i .
 - 306 5: **for** $i = 1$ to Q **do**
 - 307 6: Receive query q_i from the adversary.
 - 308 7: $J_i \leftarrow$ Sample k indices in $[L]$ with replacement.
 - 309 8: Let $a_{ij} \leftarrow \mathcal{D}_i(q_j) \in \{0, 1\}$ and $N_i := \frac{1}{k} |\{j \in J_i \mid a_{ij} = 1\}|$.
 - 310 9: Let $\hat{N}_i = N_i + \text{Lap}(\frac{1}{k})$.
 - 311 10: Output $\mathbb{1}[\hat{N}_i > \frac{1}{2}]$
-

312 To analyze this algorithm, we argue that for all $i \in [Q]$, at least $\frac{8}{10}$ of the k answers a_{ij} are correct,
 313 even in the presence of adversarially generated queries. To do this, we first need to show that
 314 the algorithm is differentially private with respect to the input random strings R . As a result, if
 315 set $L = 2400 \log^{1.5}(1/\delta) \cdot \sqrt{2Q}$ and $k = \log(Q/\delta)$ we get a robust decider that succeeds with
 316 probability at least $1 - \Theta(\delta)$. Our analysis (Theorem C.1) is included in full in Appendix C.

317 4.2 BUCKETING-BASED SEARCH

318 To perform the final search, we partition our point dataset S into n^α segments, for $\alpha < 1$. We then
 319 instantiate a copy $\mathcal{A}_i \equiv \mathcal{A}_{\text{dec}, i}$ of \mathcal{A}_{dec} in each segment. When a query comes in, we forward it to

each \mathcal{A}_i and if some segment answers $\mathbf{1}$, we perform an exhaustive search in the segment to find a point in $B_S(q, cr)$.

Algorithm 2 Robust ANN via Weak Decisions and Bucketing

- 1: **Parameters:** Error probability $\delta > 0$, number of queries Q
 - 2: Partition point set S arbitrarily into $\kappa = n^\alpha$ segments L_1, \dots, L_κ of size n/κ .
 - 3: Initialize κ independent copies $\mathcal{A}_1, \dots, \mathcal{A}_\kappa$ of \mathcal{A}_{dec} , each with $\delta' = \delta/\kappa$
 - 4: **for** $i = 1$ to Q **do**
 - 5: Receive query q_i from the adversary.
 - 6: **for** $j = 1$ to κ where $A_j(q_i) = 1$ and all $p \in L_j$ **do**
 - 7: If $p \in B_S(q, cr)$ is found, output p and proceed to the next query.
 - 8: Output \perp and proceed to the next query.
-

Lemma 4.2. *Algorithm 2 is a δ -adversarially robust algorithm for the ANN problem.*

Proof. The algorithm can only make a mistake when all the data structures reply with $\mathbf{0}$, even though there is a point $p \in B_S(q, r)$. Consider the segment L_i for which $p \in L_i$, and examine it in isolation. Because all the copies of Algorithm \mathcal{A}_{dec} are initialized independently from each other, the adversary should be able to force \mathcal{A}_i to make a mistake, which by assumption happens with probability at most $k(\delta + 1/n^{\Omega(1)})$ over all the segments via union bound. \square

We create $n^{1-\alpha}$ segments, each containing n^α points. Recall that a single copy of Algorithm \mathcal{A}_{dec} takes $\tilde{O}(n^{1+\rho}\sqrt{Q})$ pre-processing time and space, and $\tilde{O}(n^\rho)$ time and space per query. Each copy \mathcal{A}_i runs on n^α points, so for pre-processing, our algorithm uses

$$\tilde{O}\left(n^{1-\alpha}(n^\alpha)^{1+\rho}\sqrt{Q}\right) = \tilde{O}\left(n^{1+\alpha\rho}\sqrt{Q}\right)$$

bits of space for creating $n^{1-\alpha}$ copies $\mathcal{A}_1, \dots, \mathcal{A}_L$. On the other hand, to process a single query the algorithm uses

$$\tilde{O}\left(n^{1-\alpha} \cdot (n^\alpha)^\rho + n^\alpha\right) = \tilde{O}\left(n^{1-\alpha+\alpha\rho} + n^\alpha\right)$$

time. To balance the summands in the query complexity term, we set

$$n^{1-\alpha+\alpha\rho} = n^\alpha \implies \alpha = \frac{1}{2-\rho}$$

This proves Theorem 1.2 and concludes the analysis of Algorithm 2.

The biggest advantage of our algorithm is that it does not make any assumptions on the input dataset. However, it achieves sublinear query time as $\frac{1}{2-\rho} < 1$ when $\rho < 1$. Furthermore, the space complexity of our algorithm for small values of Q is superior to the space complexity of even the oblivious ANN algorithm that has space complexity $n^{1+\rho}$.

5 RELAXED FAIR ANN VIA CONCENTRIC LSH ANNULI

As a warm-up, we first present an algorithmic improvement to Theorem 3.2, removing the dependency on the ratio $\frac{n(q, cr)}{n(q, r)}$ which could grow as big as n in the query time. We achieve purely sublinear time for a relaxed fairness guarantee:

Definition 5.1 (Relaxed Fairness in ANN). *Let S be the input dataset and $q \in \mathcal{M}$ be a query point. If $B_S(q, r) \neq \emptyset$, the algorithm aims to output some point chosen uniformly at random, independently of past queries, from $B_S(q, r')$, where $r' \in [r, cr]$ is a random variable depending on q and S . Otherwise, if $B_S(q, r) = \emptyset$, the algorithm can either answer \perp or output a uniformly random point from $B_S(q, r')$ with $r' \in (r, cr]$.*

Consider the following sequence of radii between r and cr , interspersed so that the ratio between two consecutive ones is constant: $r_0 = r, r_1, \dots, r_{k-1}, r_k = cr$ are defined as $r_i = c' \cdot r_{i-1}$ for $i \in \{1, \dots, k\}$, where $c' = \sqrt[k]{c}$. We create k instances of $\mathcal{A}_{\text{fair}}$, where the i -th instance is initialized with parameters (c', r_k) . We run each instance to output a point uniformly from $B_S(q, r_i)$. If we observe an instance running for longer than $100d(n^{\rho'} + n^{1/k}) \log n$ timesteps, we stop the execution and switch to the next instance.

Claim 5.2. Consider a query q and suppose $B_S(q, r) \neq \emptyset$. There exists $i \in \{0, \dots, k-1\}$ such that:

$$\frac{n(q, r_{i+1})}{n(q, r_i)} \leq n^{\frac{1}{k}} \quad (2)$$

Proof. Since $n(q, r) \geq 1$ it also holds that $n(q, r_i) \geq 1$ for all $i \in \{0, \dots, k\}$. Suppose that for all $i \in \{0, \dots, k-1\}$ it holds that $\frac{n(q, r_{i+1})}{n(q, r_i)} > n^{\frac{1}{k}}$. Then, via a telescoping product we arrive at a contradiction:

$$\frac{n(q, cr)}{n(q, r)} = \frac{n(q, r_1)}{n(q, r_0)} \cdot \frac{n(q, r_3)}{n(q, r_2)} \dots \frac{n(q, r_{k-1})}{n(q, r_{k-2})} \cdot \frac{n(q, cr)}{n(q, r_{k-1})} > \left(n^{\frac{1}{k-1}}\right)^{k-1} = n \quad \square$$

Claim 5.2 shows that if $B_S(q, r) \neq \emptyset$ we output a uniformly sampled point from some $B_S(q, r_i)$, where r_i is a random variable R depending on S, q and our algorithm's randomness. On the other hand, if $B_S(q, r) = \emptyset$, we either output \perp if all the copies \mathcal{D}_i time-out, or a uniformly sampled point from some sphere $B_S(q, r_i)$. In either case, we enjoy the relaxed fairness guarantee of Definition 5.1. For the runtime, our algorithm takes space $O(kn^{1+\rho(c)})$ for pre-processing, and time $\tilde{O}(dk \cdot \min\{n^{\rho(c)}, n^{1/k}\})$ for answering each query:

Theorem 5.3. There exists an algorithm for solving the relaxed fair (c, r) -ANN problem that uses $\tilde{O}(dn^\beta)$ time per query and $\tilde{O}(n^{1+\beta})$ time for pre-processing, where $\beta = \min_{k \in \mathbb{Z}} \max\{\rho(c^{1/k}), 1/k\}$.

Solving for β is metric space dependent. For the hypercube, we can use $\rho(c) = \frac{1}{2c-1}$, a back-of-the-envelope calculation yields $k = \Theta\left(\frac{\log c}{\log \log c}\right)$. To nail down the constants precisely, we pick:

$$\beta = \min \left\{ \max \left\{ \frac{1}{\lfloor k^* \rfloor}, \frac{1}{2c^{1/\lfloor k^* \rfloor} - 1} \right\}, \max \left\{ \frac{1}{\lceil k^* \rceil}, \frac{1}{2c^{1/\lceil k^* \rceil} - 1} \right\} \right\}$$

with our algorithm having runtime $\tilde{O}(dn^\beta)$ and space complexity $\tilde{O}(\sqrt{Q} \cdot n^{1+\beta})$. For instance, if $c = 4$ we have $k^* = 2.48$, so $\beta = 1/3$, while for $c = 10$ we have $k^* \approx 3.15$ so $\beta = 1/3$.

We plot the solutions for β for $c \in [2, 100]$ in Figure 1. Note that $\beta \rightarrow 0$ as $c \rightarrow \infty$.

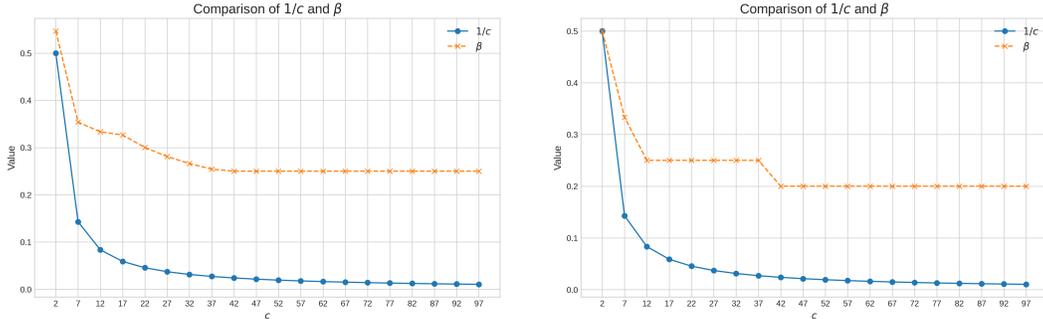


Figure 1: Solutions for β for different values of c in the hypercube (left) and ℓ_2 (right) domains.

6 ROBUST ANN IMPROVEMENTS

We now combine our concentric annuli technique with fair ANN to develop a more efficient and robust algorithm.

We again partition the space into $k \geq 1$ concentric annuli $(r_{i-1}, r_i]$, where $r_0 = r$ and $r_i = c' \cdot r_{i-1}$ for $c' = \sqrt[k]{c}$. For each annulus i , we instantiate two independent copies of the base algorithm: a testing instance $\mathcal{A}_i \leftarrow \mathcal{A}_{\text{fair}}(c', r_{i-1})$ and a held-out execution instance $\mathcal{A}_{\text{fair}}^{(i)} \leftarrow \mathcal{A}_{\text{fair}}(c', r_{i-1})$.

Our goal is to find an annulus whose algorithm runs quickly. We formalize this notion as follows.

Definition 6.1. Let T_i be the random runtime of the testing instance \mathcal{A}_i . The i -th annulus is a **good annulus** if its probability of fast termination, p_i , is high:

$$p_i := \Pr[T_i \leq 4d(n^{\rho(c')} + n^{1/k}) \log n] \geq 0.999$$

Upon receiving a query q , we estimate each probability p_i with an additive error of at most η by observing the fraction of $\Theta(\eta^{-2} \log(kQn))$ independent sub-trials of \mathcal{A}_i that halt within the time bound. Let \hat{p}_i be this empirical estimate for p_i . We identify candidate annuli with an indicator vector $\hat{\alpha} \in \{0, 1\}^k$, where: $\hat{\alpha}_i = \mathbb{1}[\hat{p}_i \geq 0.997]$.

With high probability, $\hat{\alpha}_i = 1$ implies that annulus i is good. A similar argument to Claim 5.2 guarantees that at least one good annulus must exist. We therefore find the first index i^* for which $\hat{\alpha}_{i^*} = 1$ and run the corresponding execution instance $\mathcal{A}_{\text{fair}}^{(i^*)}$ to completion. This approach yields a solution in $\tilde{O}(d(n^{\rho(c')} + n^{1/k}))$ time with probability at least 0.998

To ensure robustness, the release of the vector $\hat{\alpha}$ (and thus the choice of i^*) must not reveal information about the internal randomness of our algorithm instances. We therefore use the DP-based robustification framework of Hassidim et al. (2022) to release $\hat{\alpha}$ privately. While this increases the space complexity by a factor of \sqrt{Q} , it allows us to achieve a considerably better, assumption-free, and purely sublinear query time. Algorithm 3 presents the details of our approach, and the full analysis, as well as the proof of Theorem 1.3, can be found in Appendix D.

Algorithm 3 Improved Robust ANN Search

```

1: Input: Query  $q \in \mathcal{M}$ , parameters  $c, r, k \geq 1$  and  $\delta \in (0, 0.995)$ 
2: procedure INITIALIZE
3:   Let  $c' \leftarrow \sqrt[k]{c}$  and  $r_0 \leftarrow r$ .
4:   Let  $\eta = 0.001$ ,  $m = \eta^{-2} \log(Qk/\delta)$  and  $L = 2400 \log^{1.5}(1/\delta) \sqrt{2Q}$ .
5:   for  $i = 1, \dots, k$  do
6:     Let  $N = L \cdot m$  and  $r_i \leftarrow c' \cdot r_{i-1}$ .
7:     Instantiate  $N$  copies  $\mathcal{A}_{i,j} \leftarrow \mathcal{A}_{\text{fair}}(c', r_{i-1})$ . ▷ Testing Instances ( $m \times L$  grid)
8:     Instantiate  $\mathcal{A}_{\text{fair}}^{(i)} \leftarrow \mathcal{A}_{\text{fair}}(c', r_{i-1})$  ▷ Execution Instances
9: procedure QUERY( $q$ )
10:  for  $i \in \{1, \dots, k\}$  do
11:    Let  $S_{\text{trunc}} \leftarrow 4d(n^{1/k} + n^{\rho(c')}) \log n$  ▷ Let  $p_j \leftarrow \Pr[T_j < S_{\text{trunc}}]$ .
12:     $J_i \leftarrow$  Sample  $s = \log(Qk/\delta)$  indices in  $[L]$  with replacement.
13:    for  $j \in J_i$  do
14:      Let  $\tilde{p}_{ij} \leftarrow \mathbb{1}[T_{i,jm+w} < S_{\text{trunc}}]$  for  $w \in \{0, \dots, m-1\}$ .
15:    Let  $\hat{p}_i \leftarrow \frac{1}{s} \sum_{j \in J_i} \tilde{p}_{ij} + \text{Lap}(\frac{1}{s})$ .
16:    Set  $\hat{a}(q)_i \leftarrow \mathbb{1}[\hat{p}_i \geq 0.998]$ .
17:    if  $\hat{a}(q) = \vec{0}$  then
18:      return  $\perp$ 
19:    else
20:       $i^* \leftarrow \min\{i \in \{1, \dots, k\} \mid \hat{a}(q)_i = 1\}$  ▷ Find most significant bit index
21:    return  $\mathcal{A}_{\text{fair}}^{(i^*)}(q)$ 

```

7 CONCLUSION

This study presents a series of algorithms for solving ANN against adaptive adversaries. Our approaches, which integrate principles of fairness and privacy with novel data constructions, offers are efficient and input-independent. Our work raises several intriguing questions for future research: Can we establish time and space lower bounds for robust algorithms? How do our algorithms perform against adversaries with more information, such as the timestamps? Can the powerful link between fairness and robustness be extended to other domains, like estimation problems? Lastly, a practical implementation of our approach and the nuances it presents in a real system are also important avenues to investigate. We believe this work provides a strong foundation for future exploration into these areas.

REFERENCES

- 486
487
488 Alexandr Andoni. *Nearest neighbor search: the old, the new, and the impossible*. PhD thesis,
489 Massachusetts Institute of Technology, 2009.
- 490 Alexandr Andoni and Daniel Beaglehole. Learning to hash robustly, guaranteed. In *International*
491 *Conference on Machine Learning*, pp. 599–618. PMLR, 2022.
- 492 Alexandr Andoni and Piotr Indyk. Nearest neighbors in high-dimensional spaces. In *Handbook of*
493 *Discrete and Computational Geometry*, pp. 1135–1155. Chapman and Hall/CRC, 2017.
- 494 Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near
495 neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*,
496 pp. 793–801, 2015.
- 497
498 Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Lower bounds on time-
499 space trade-offs for approximate near neighbors. *arXiv preprint arXiv:1605.02701*, 2016.
- 500 Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal hashing-based
501 time-space trade-offs for approximate near neighbors. In *Proceedings of the twenty-eighth annual*
502 *ACM-SIAM symposium on discrete algorithms*, pp. 47–66. SIAM, 2017a.
- 503 Alexandr Andoni, Ilya Razenshteyn, and Negev Shekel Nosatzki. Lsh forest: Practical algorithms
504 made theoretical. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Dis-*
505 *crete Algorithms*, pp. 67–78. SIAM, 2017b.
- 506 Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high
507 dimensions. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro*
508 *2018*, pp. 3287–3318. World Scientific, 2018.
- 509
510 Idan Attias, Edith Cohen, Moshe Shechner, and Uri Stemmer. A framework for adversarial stream-
511 ing via differential privacy and difference estimators. *Algorithmica*, pp. 1–56, 2024.
- 512
513 Martin Aumüller, Sariel Har-Peled, Sepideh Mahabadi, Rasmus Pagh, and Francesco Silvestri. Sam-
514 pling a near neighbor in high dimensions—who is the fairest of them all? *ACM Transactions on*
515 *Database Systems (TODS)*, 47(1):1–40, 2022.
- 516 Raef Bassily, Adam Smith, Thomas Steinke, and Jonathan Ullman. More general queries and less
517 generalization error in adaptive data analysis. *arXiv preprint arXiv:1503.04843*, 2015.
- 518 Raef Bassily, Kobbi Nissim, Adam Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman.
519 Algorithmic stability for adaptive data analysis. In *Proceedings of the forty-eighth annual ACM*
520 *symposium on Theory of Computing*, pp. 1046–1059, 2016.
- 521
522 MohammadHossein Bateni, Laxman Dhulipala, Willem Fletcher, Kishen N Gowda, D Ellis Her-
523 shkowitz, Rajesh Jayaram, and Jakub Lacki. Efficient centroid-linkage clustering. *Advances in*
524 *Neural Information Processing Systems*, 37:49649–49683, 2024.
- 525
526 Soheil Behnezhad, Rajmohan Rajaraman, and Omer Wasim. Fully dynamic $(\Delta + 1)$ -coloring against
527 adaptive adversaries. In Yossi Azar and Debmalya Panigrahi (eds.), *Proceedings of the 2025*
528 *Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA,*
529 *January 12-15, 2025*, pp. 4983–5026. SIAM, 2025. doi: 10.1137/1.9781611978322.169. URL
530 <https://doi.org/10.1137/1.9781611978322.169>.
- 531 Amos Beimel, Haim Kaplan, Yishay Mansour, Kobbi Nissim, Thatchaphol Saranurak, and Uri
532 Stemmer. Dynamic algorithms against an adaptive adversary: generic constructions and lower
533 bounds. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*,
534 pp. 1671–1684, 2022.
- 535 Omri Ben-Eliezer, Talya Eden, and Krzysztof Onak. Adversarially robust streaming via dense-
536 sparse trade-offs. In *Symposium on Simplicity in Algorithms (SOSA)*, pp. 214–227. SIAM, 2022a.
- 537 Omri Ben-Eliezer, Rajesh Jayaram, David P Woodruff, and Eylon Yogev. A framework for adver-
538 sarily robust streaming algorithms. *ACM Journal of the ACM (JACM)*, 69(2):1–33, 2022b.
- 539

- 540 Liron Bergman, Niv Cohen, and Yedid Hoshen. Deep nearest neighbor anomaly detection. *arXiv*
541 *preprint arXiv:2002.10445*, 2020.
542
- 543 Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. Min-wise inde-
544 pendent permutations. In *Proceedings of the thirtieth annual ACM symposium on Theory of*
545 *computing*, pp. 327–336, 1998.
- 546 Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Differentially private release and learn-
547 ing of threshold functions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer*
548 *Science*, pp. 634–649. IEEE, 2015.
549
- 550 Amit Chakrabarti and Manuel Stoeckl. Finding missing items requires strong forms of randomness.
551 In *39th Computational Complexity Conference (CCC 2024)*. Schloss Dagstuhl–Leibniz-Zentrum
552 für Informatik, 2024.
- 553 Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Adversarially robust coloring for graph
554 streams. *arXiv preprint arXiv:2109.11130*, 2021.
555
- 556 Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of*
557 *the thirty-fourth annual ACM symposium on Theory of computing*, pp. 380–388, 2002.
558
- 559 Yeshwanth Cherapanamjeri and Jelani Nelson. On adaptive distance estimation. *Advances in Neural*
560 *Information Processing Systems*, 33:11178–11190, 2020.
- 561 Yeshwanth Cherapanamjeri and Jelani Nelson. Terminal embeddings in sublinear time. *TheoretCS*,
562 3, 2024.
563
- 564 Yeshwanth Cherapanamjeri, Sandeep Silwal, David P Woodruff, Fred Zhang, Qiuyi Zhang, and
565 Samson Zhou. Robust algorithms on adaptive inputs from bounded adversaries. *arXiv preprint*
566 *arXiv:2304.07413*, 2023.
- 567 Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing
568 scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on*
569 *Computational geometry*, pp. 253–262, 2004.
570
- 571 Itai Dinur, Uri Stemmer, David P Woodruff, and Samson Zhou. On differential privacy and adap-
572 tive data analysis with bounded space. In *Annual International Conference on the Theory and*
573 *Applications of Cryptographic Techniques*, pp. 35–65. Springer, 2023.
- 574 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity
575 in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference,*
576 *TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pp. 265–284. Springer, 2006.
577
- 578 Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010*
579 *IEEE 51st annual symposium on foundations of computer science*, pp. 51–60. IEEE, 2010.
580
- 581 Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toni Pitassi, Omer Reingold, and Aaron Roth. Gen-
582 eralization in adaptive data analysis and holdout reuse. *Advances in neural information processing*
583 *systems*, 28, 2015a.
- 584 Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon
585 Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh*
586 *annual ACM symposium on Theory of computing*, pp. 117–126, 2015b.
- 587 Shiyuan Feng, Ying Feng, George Zhaoqi Li, Zhao Song, David Woodruff, and Lichen Zhang.
588 On differential privacy for adaptively solving search problems via sketching. In *Forty-second*
589 *International Conference on Machine Learning*, 2025.
590
- 591 Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hash-
592 ing. In *Proceedings of the 25th International Conference on Very Large Data Bases, VLDB*
593 *’99*, pp. 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN
1558606157.

- 594 Elena Gribelyuk, Honghao Lin, David P Woodruff, Huacheng Yu, and Samson Zhou. A strong
595 separation for adversarially robust ℓ_0 estimation for linear sketches. In *2024 IEEE 65th Annual*
596 *Symposium on Foundations of Computer Science (FOCS)*, pp. 2318–2343. IEEE, 2024.
- 597
598 Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David P Woodruff, and Amir Zandieh.
599 Hyperattention: Long-context attention in near-linear time. In *International Conference on Learn-*
600 *ing Representations (ICLR)*, 2024. arXiv preprint arXiv:2310.05869.
- 601 Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards re-
602 moving the curse of dimensionality. *Theory of Computing*, 1(8):321–350, 2012.
- 603
604 Moritz Hardt and David P Woodruff. How robust are linear sketches to adaptive inputs? In *Pro-*
605 *ceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 121–130, 2013.
- 606 Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially
607 robust streaming algorithms via differential privacy. *Journal of the ACM*, 69(6):1–14, 2022.
- 608 Jeffrey Ichnowski and Ron Alterovitz. Fast nearest neighbor search in $se(3)$ for sampling-based mo-
609 tion planning. In *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh*
610 *International Workshop on the Algorithmic Foundations of Robotics*, pp. 197–214. Springer, 2015.
- 611
612 Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of
613 dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*,
614 pp. 604–613, 1998.
- 615 Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Cru-
616 shev. A survey on locality sensitive hashing algorithms and their applications. *arXiv preprint*
617 *arXiv:2102.08942*, 2021.
- 618
619 Yannis Kalantidis and Yannis Avrithis. Locally optimized product quantization for approximate
620 nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern*
621 *recognition*, pp. 2321–2328, 2014.
- 622 Michael Kapralov, Mikhail Makarov, and Christian Sohler. On the adversarial robustness of locality-
623 sensitive hashing in hamming space. *arXiv preprint arXiv:2402.09707*, 2024.
- 624
625 Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer.
626 In *International Conference on Learning Representations (ICLR)*, 2020. arXiv preprint
627 arXiv:2001.04451.
- 628
629 Lifeng Lai and Erhan Bayraktar. On the adversarial robustness of robust estimators. *IEEE Transac-*
630 *tions on Information Theory*, 66(8):5097–5109, 2020.
- 631
632 Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. *ACM Computing Surveys*
633 *(CSUR)*, 28(1):33–37, 1996.
- 634
635 Ninh Pham and Rasmus Pagh. Scalability and total recall with fast coveringlsh. In *Proceedings*
636 *of the 25th ACM International on Conference on Information and Knowledge Management*, pp.
637 1109–1118, 2016.
- 638
639 John SantaLucia, Hatim T Allawi, and P Ananda Seneviratne. Improved nearest-neighbor param-
640 eters for predicting dna duplex stability. *Biochemistry*, 35(11):3555–3562, 1996.
- 641
642 Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algo-*
643 *rithms*. Cambridge university press, 2014.
- 644
645 Adam Smith. Information, privacy and stability in adaptive data analysis. *arXiv preprint*
646 *arXiv:1706.00820*, 2017.
- 647
648 Manuel Stoeckl. Streaming algorithms for the missing item finding problem. In *Proceedings of the*
649 *2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 793–818. SIAM, 2023.
- 650
651 Yukihiro Tagami. Annexml: Approximate nearest neighbor search for extreme multi-label classifica-
652 tion. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery*
653 *and data mining*, pp. 455–464, 2017.

648 Koen Verstrepen and Bart Goethals. Unifying nearest neighbors collaborative filtering. In *Proceed-*
649 *ings of the 8th ACM Conference on Recommender systems*, pp. 177–184, 2014.

650

651 Alexander Wei. Optimal las vegas approximate near neighbors in ℓ_p . *ACM Transactions on Algo-*
652 *rithms (TALG)*, 18(1):1–27, 2022.

653 David P Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding
654 windows via difference estimators. In *2021 IEEE 62nd Annual Symposium on Foundations of*
655 *Computer Science (FOCS)*, pp. 1183–1196. IEEE, 2022.

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702	APPENDIX CONTENTS	
703		
704		
705	A Related and Concurrent Work	14
706	A.1 Comparison with Feng et al. (2025)	15
707		
708	B Review of Differential Privacy	15
709		
710	B.1 Definition of differential privacy	15
711	B.2 The Laplace Mechanism and its properties	15
712		
713	B.3 Properties of differential privacy	16
714		
715	C Proof of Theorem C.1	16
716		
717		
718	D Proof of Theorem 1.3	18
719		
720	E Improved Robust ANNS Algorithms with \forall guarantees	20
721		
722	E.1 A <i>For-all</i> guarantee in the Hamming cube	20
723	E.1.1 Improving the query runtime via sampling	21
724	E.1.2 Utilizing the optimal LSH algorithm	22
725		
726	E.2 Discretization of continuous spaces through metric coverings	23
727		
728	E.2.1 Metric coverings in continuous spaces	23
729	E.2.2 The robust ANN algorithm	23
730	E.2.3 Removing the dependency on the scale	25
731		

733 A RELATED AND CONCURRENT WORK

734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750

The challenge of designing algorithms robust to adversarial queries is well-studied, particularly in privacy and statistics (Bassily et al., 2015; Smith, 2017; Bassily et al., 2016), where Differential Privacy is a central tool for ensuring robustness (Dwork et al., 2015a; Dinur et al., 2023). The question of adversarial robustness was formally introduced to streaming algorithms by Ben-Eliezer et al. (Ben-Eliezer et al., 2022b), motivated by attacks on linear sketches (Hardt & Woodruff, 2013), and has since inspired a long line of work on robustifying various streaming algorithms (Hassidim et al., 2022; Chakrabarti et al., 2021; Lai & Bayraktar, 2020; Chakrabarti & Stoeckl, 2024; Stoeckl, 2023; Woodruff & Zhou, 2022; Ben-Eliezer et al., 2022a).

743
744
745
746
747
748
749
750

Our work is most directly inspired by the framework of Hassidim et al. (Hassidim et al., 2022), who used Differential Privacy to solve estimation problems robustly, and by Cherapanamjeri et al. (Cherapanamjeri et al., 2023), who applied this framework with low query time overhead. While we adapt a similar approach, their methods are fundamentally limited to estimation and don’t extend to search problems like NNS, where the output must be a specific dataset element. The difficulty of robust search is further highlighted by Beimel et al. (Beimel et al., 2022), who established lower bounds showing that robust algorithms for certain search problems are inherently slower than their oblivious counterparts, motivating our investigation.

751
752
753
754
755

Different works further reinforce the unique challenges of robust search. Work on robust graph coloring, for example, also requires techniques beyond simple noise addition due to its discrete output space (Chakrabarti et al., 2021; Behnezhad et al., 2025). Our approach is also distinct from Las Vegas LSH constructions (Pham & Pagh, 2016; Wei, 2022). While these methods guarantee no false negatives, they remain vulnerable to adversaries who can inflate their expected runtime (Kapralov et al., 2024). Our focus, in contrast, is on robustifying traditional Monte Carlo algorithms.

Finally, our approach builds on the use of discretization and net-based arguments to achieve ‘for-all’ guarantees for ANN. This technique was previously used for robust distance estimation (Cherapanamjeri & Nelson, 2020), for ANN in conjunction with partition trees (Cherapanamjeri & Nelson, 2024) and for efficient centroid-linkage clustering (Bateni et al., 2024). We contribute a simpler and more streamlined construction that offers a modest performance improvement over this prior work.

A.1 COMPARISON WITH FENG ET AL. (2025)

Our work was developed concurrently and independently with Feng et al. (2025). Our approaches, assumptions, and performance guarantees differ significantly.

Methodology Feng et al. (2025) propose a method tightly coupled to the structure of DP noise via a reduction to the private selection problem. In contrast, our “search-to-decision” and fairness frameworks are more general, treating the DP component as a black-box primitive.

Assumptions Their algorithm’s complexity depends on a near-neighbor density bound s , where $|B_S(q, cr)| \leq s$. We present the first algorithms whose query runtimes are independent of the input dataset, making them robust to worst-case data distributions.

Performance Their query time scales *multiplicatively* with the number of points in the annulus, $|B_S(q, cr)|$, while our algorithms are either purely sublinear or their query time depends *additively* only on the density ratio $D = \frac{|B_S(q, cr)|}{|B_S(q, r)|}$. Crucially, this dependency on D does not affect our space complexity, which still grows by an additional factor of \sqrt{Q} .

B REVIEW OF DIFFERENTIAL PRIVACY

Our work leans heavily on results from differential privacy, so we give the necessary definitions and results here.

B.1 DEFINITION OF DIFFERENTIAL PRIVACY

Definition B.1 (Differential Privacy). Let \mathcal{A} be any randomized algorithm that operates on databases whose elements come from some universe. For parameters $\varepsilon > 0$ and $\delta \in [0, 1]$, the algorithm \mathcal{A} is (ε, δ) -differentially private (DP) if for any two neighboring databases $S \sim S'$ (ones that differ on one row only), the distributions on the algorithm’s outputs when run on S vs S' are very close. That is, for any $S \sim S'$ and any subset of outcomes T of the output space of \mathcal{A} we have:

$$\Pr[\mathcal{A}(S) \in T] \leq e^\varepsilon \cdot \Pr[\mathcal{A}(S') \in T] + \delta$$

B.2 THE LAPLACE MECHANISM AND ITS PROPERTIES

Theorem B.2 (The Laplace Mechanism, (Dwork et al., 2006)). Let $f : X^* \rightarrow \mathbb{R}$ be a function. Define its sensitivity ℓ to be an upper bound to how much f can change on neighboring databases:

$$\forall S \sim S' : |f(S) - f(S')| \leq \ell$$

The algorithm that on input $S \in X^*$ returns $f(S) + \text{Lap}\left(\frac{\ell}{\varepsilon}\right)$ is $(\varepsilon, 0)$ -DP, where

$$\text{Lap}(\lambda; x) := \frac{1}{2\lambda} \exp\left(-\frac{|x|}{\lambda}\right)$$

is the Laplace Distribution over \mathbb{R} .

We will make use of the following concentration property of the Laplace Distribution:

Lemma B.3. For $m \geq 1$, let $Z_1, \dots, Z_m \sim \text{Lap}(\lambda)$ be iid random variables. We have that:

$$\Pr\left[\max_{i=1}^m Z_i > \lambda(\ln(m) + t)\right] \leq e^{-t}$$

B.3 PROPERTIES OF DIFFERENTIAL PRIVACY

Differential Privacy has numerous properties that are useful in the design of algorithms. The following theorem is known as “advanced adaptive composition” and describes a situation when DP algorithms are linked sequentially in an adaptive way.

Theorem B.4 (Advanced Composition, (Dwork et al., 2010)). *Suppose algorithms $\mathcal{A}_1, \dots, \mathcal{A}_k$ are (ε, δ) -DP. Let \mathcal{A}' be the adaptive composition of these algorithms: on input database x , algorithm \mathcal{A}_i is provided with x , and, for $i \geq 2$, with the output y_{i-1} of \mathcal{A}_{i-1} . Then, for any $\delta' \in (0, 1)$, Algorithm \mathcal{A} is $(\tilde{\varepsilon}, \tilde{\delta})$ -DP with:*

$$\tilde{\varepsilon} = \varepsilon \cdot \sqrt{2k \ln(1/\delta')} + 2k\varepsilon^2 \text{ and } \tilde{\delta} = k\delta + \delta'$$

There is also a composition theorem concerning situations where the dataset is partitioned:

Theorem B.5 (Parallel Composition). *Let f_1, \dots, f_k be $(\varepsilon, 0)$ -DP mechanisms and X be a dataset. Suppose X is partitioned into k parts X_1, \dots, X_k and let $f(X) = (f_1(X_1), \dots, f_k(X_k))$. Then f is $(\varepsilon, 0)$ -DP.*

The next theorem dictates that post-processing the output of a DP algorithm cannot degrade its privacy guarantees, as long as the processing does not use information from the original database.

Theorem B.6 (DP is closed under Post-Processing). *Let $\mathcal{A} : U^n \rightarrow Y^m$ and $\mathcal{B} : Y^m \rightarrow Z^r$ be randomized algorithms, where U, Y, Z are arbitrary sets. If \mathcal{A} is (ε, δ) -DP, then so is the composed algorithm $\mathcal{B}(\mathcal{A}(\cdot))$.*

The following theorem showcases the power of DP algorithms in learning.

Theorem B.7 (DP and Generalization, (Bassily et al., 2016; Dwork et al., 2015b)). *Let $\varepsilon \in (0, 1/3)$ and $\delta \in (0, \varepsilon/4)$. Let \mathcal{A} be a (ε, δ) -DP algorithm that operates on databases in X^n and outputs m predicate functions $h_i : X \rightarrow \{0, 1\}$ for $i \in [m]$. Then, if D is any distribution over X and S consists of $n \geq \frac{1}{\varepsilon^2} \cdot \log\left(\frac{2\varepsilon m}{\delta}\right)$ iid samples from D , we have for all $i \in [m]$ that:*

$$\Pr_{\substack{S \sim D^n \\ h_i \leftarrow \mathcal{A}(S)}} \left[\left| \frac{1}{|S|} \sum_{x \in S} h_i(x) - \mathbb{E}_{x \sim D} [h_i(x)] \right| \geq 10\varepsilon \right] \leq \frac{\delta}{\varepsilon}$$

In other words, a privately generated predicate is a good estimator of its expectation under any distribution on the input data. A final property of privacy that we will use is a boosting technique through sub-sampling:

Theorem B.8 (Privacy Amplification by Subsampling, (Bun et al., 2015; Cherapanamjeri et al., 2023)). *Let \mathcal{A} be an (ε, δ) -DP algorithm operating on databases of size m . For $n \geq 2m$, consider an algorithm that for input a database of size n , it subsamples (with replacement) m rows from the database and runs \mathcal{A} on the result. Then this algorithm is (ε', δ') -DP for*

$$\varepsilon' = \frac{6\varepsilon m}{n} \text{ and } \delta' = \exp\left(\frac{6\varepsilon m}{n}\right) \cdot \frac{4m}{n} \cdot \delta$$

C PROOF OF THEOREM C.1

In this section we include a formal analysis of the construction of Algorithm 1. We prove the following theorem:

Theorem C.1. *Let \mathcal{A} be an oblivious decider algorithm for ANN that uses $s(n)$ space and $t(n)$ time per query. Let $\delta \in (0, 0.995)$ and suppose we set $L = 2400 \log^{1.5}(1/\delta) \cdot \sqrt{2Q}$ and $k = \log(Q/\delta)$. Then, the algorithm \mathcal{A}_{dec} is an adversarially robust decider that succeeds with probability at least $1 - \Theta(\delta)$ using $s(n) \cdot \tilde{O}(\sqrt{Q})$ bits of space and $\tilde{O}(t(n))$ time per query.*

First, we show that the algorithm is differentially private with respect to its input randomness.

Lemma C.2. *Let $\varepsilon = 0.01$ and $\delta \in (0, 0.995)$. Algorithm \mathcal{A}_{dec} is (ε, δ) -DP with respect to the string of randomness R .*

864 *Proof.* We analyze the privacy of the algorithm \mathcal{A}_{dec} given in Algorithm 1 with respect to the string
865 of randomness R , which we interpret as its input. Suppose we let

$$866 \quad \varepsilon' = \frac{\varepsilon}{2\sqrt{2Q \ln(1/\delta)}}$$

867 For all $i \in [Q]$, we claim that the response to query q_i is $(\varepsilon', 0)$ -DP with respect to R . This is because
868 the statistic N_i defined in Line 8 of Algorithm 1 has sensitivity $1/k$ and therefore by Theorem B.2,
869 after applying the Laplace mechanism in Line 9, we have that releasing \widehat{N}_i is $(1, 0)$ -DP with respect
870 to the strings R . The binary output based on comparing \widehat{N}_i with the constant threshold $1/2$ is still
871 $(1, 0)$ -DP by post-processing (Theorem B.6).

872 Since $L \geq 2k$, using the amplification by sub-sampling property (Theorem B.8), we get that each
873 iteration is $(\varepsilon', 0)$ -DP, because for large enough Q we have:

$$874 \quad \frac{6k}{L} = \frac{6\varepsilon \log \frac{1}{\delta} + 6\varepsilon \log Q}{24 \cdot \log \frac{1}{\delta} \sqrt{2Q \ln \left(\frac{1}{\delta}\right)}} < \frac{2\varepsilon}{4\sqrt{2Q \ln \left(\frac{1}{\delta}\right)}} = \varepsilon'$$

875 Finally, by adaptive composition (Theorem B.4), after Q adaptive steps our resulting algorithm is
876 (ε'', δ) -DP where:

$$877 \quad \varepsilon'' = \varepsilon' \sqrt{2Q \ln \left(\frac{1}{\delta}\right)} + Q(\varepsilon')^2 = \frac{\varepsilon}{2} + \frac{\varepsilon^2}{4 \ln \left(\frac{1}{\delta}\right)} \leq \varepsilon$$

878 for $\varepsilon \leq 2 \ln \delta^{-1}$, which is satisfied for $\delta \in (0, 0.995)$. Thus, Algorithm \mathcal{A}_{dec} is (ε, δ) -DP with
879 respect to its inputs – the random strings R . \square

880 Next, we show that a majority of the data structures \mathcal{D}_i output accurate verdicts with high probability,
881 even against adversarially generated queries.

882 **Lemma C.3.** *With probability at least $1 - \delta$, for all $i \in [Q]$, at least $0.8L$ of the answers a_{ij} are
883 accurate responses to the decision problem with query q_i .*

884 *Proof.* The central idea of the proof, as it appeared in (Hassidim et al., 2022), is to imagine the
885 adversary \mathcal{B} as a post-processing mechanism that tries to guess which random strings lead \mathcal{A} to
886 making a mistake.

887 Imagine a wrapper *meta-algorithm* \mathcal{C} , outlined as Algorithm 4, that takes as input the random string
888 $R = r_1 \circ r_2 \circ \dots \circ r_L$, which is generated according to some unknown, arbitrary distribution \mathcal{R} . This
889 algorithm \mathcal{C} simulates the game between \mathcal{A}_{dec} and \mathcal{B} : It first runs \mathcal{B} to provide some input dataset
890 $S \subseteq U$ to \mathcal{A}_{dec} , which is seeded with random strings in R . Then, \mathcal{C} uses \mathcal{B} to query \mathcal{A}_{dec} adaptively
891 with queries (q_1, \dots, q_Q) . At the same time, it simulates \mathcal{A}_{dec} to receive answers a_1, \dots, a_Q that are
892 fed back to \mathcal{B} . By Lemma C.2, the output (a_1, \dots, a_Q) is produced privately with respect to R ,
893 regardless of how the adversary makes their queries.

894 At every step i , once \mathcal{B} has provided $\vec{q}_i = (q_1, \dots, q_i)$ and has gotten back i answers (a_1, \dots, a_i) from
895 \mathcal{A}_{dec} , our meta-algorithm \mathcal{C} *post-processes* this output history $\{(q_j, a_j)\}_{j=1}^i$ to generate a predicate
896 $h_{\vec{q}_i} : \{0, 1\}^* \rightarrow \{0, 1\}$. This predicate tells which strings $r \in \{0, 1\}^*$ lead algorithm \mathcal{A} to success-
897 fully answer query prefix \vec{q}_i on input dataset S , in the decision-problem regime. More formally⁴:

$$898 \quad h_{\vec{q}_i}(r) := \bigwedge_{1 \leq j \leq i} \{\mathcal{A}(r)(S, q_j) = \mathbb{1}[B_S(q_j, \vec{r}) \neq \emptyset]\} \quad (3)$$

899 Generating these predicates is possible because $h_{\vec{q}_i}$ only depends on \vec{q}_i , which is a substring of the
900 output history that \mathcal{C} has access to. As a result, \mathcal{C} can produce $h_{\vec{q}_i}$ by (say) calculating its value for
901 each value of R exhaustively⁵. Because \mathcal{C} is only allowed to post-process the query/answer vector
902 $(q_1, a_1, \dots, q_i, a_i)$, the output predicate $h_{\vec{q}_i}$ is also generated in a (ε, δ) -DP manner with respect to
903 r_1, \dots, r_L , by Theorem B.6.

904 ⁴We replace the radius parameter r with \vec{r} briefly in this argument. The symbol r is reserved for an arbitrary
905 random string.

906 ⁵We assume \mathcal{C} has unbounded computational power.

Algorithm 4 The meta-algorithm \mathcal{C} , ran for i steps

- 1: **Inputs:** Random string $R = r_1 \circ r_2 \circ \dots \circ r_L$, descriptions of Algorithms \mathcal{A}_{dec} and \mathcal{B} .
 - 2: Simulate \mathcal{B} to obtain a dataset $S \subset U$.
 - 3: Initialize \mathcal{A}_{dec} with random strings (r_1, \dots, r_L) and the dataset S .
 - 4: **for** $i \in [Q]$ **do**
 - 5: Simulate \mathcal{B} to produce a query q_j based on the prior history of queries and answers.
 - 6: Simulate \mathcal{A} on query q_j to produce an answer.
 - 7: Compute (via post-processing of query/answer history) predicate $h_{\vec{q}_i}(\cdot)$ from Equation 3.
 - 8: **Output** $(h_{\vec{q}_1}, \dots, h_{\vec{q}_Q})$.
-

Given these Q privately generated predicates, and since $L > \frac{1}{\varepsilon^2} \log \frac{2\varepsilon Q}{\delta}$ for large enough Q , by the generalization property of DP (Theorem B.7) we have that with probability at least $1 - \frac{\delta}{\varepsilon} = 1 - \Theta(\delta)$ it holds for any distribution \mathcal{R} and for all $i \in [Q]$ that:

$$\left| \mathbb{E}_{r \sim \mathcal{R}} [h_{\vec{q}_i}(r)] - \frac{1}{L} \sum_{j=1}^L h_{\vec{q}_i}(r_j) \right| \leq 10\varepsilon = \frac{1}{10} \quad (4)$$

But if \mathcal{R} is the uniform distribution, then $\mathbb{E}_{r \sim \mathcal{R}} [h_{\vec{q}_i}(r)]$ is simply the probability that \mathcal{A}_2 gives an accurate answer on the *fixed* query sequence \vec{q}_i . Since \mathcal{A} is an oblivious decider, Equation 4 implies that:

$$\mathbb{E}_{r \sim \mathcal{R}} [h_{\vec{q}_i}(r)] \geq \frac{9}{10} \quad (5)$$

Further, $\frac{1}{L} \sum_{j=1}^L h_{\vec{q}_i}(r_j)$ is the fraction of random strings that lead \mathcal{A}_2 to be correct. Thus, by Equation 5, this fraction is at least $(\frac{9}{10} - \frac{1}{10})L = 0.8L$ for all $i \in [Q]$. \square

We are now ready to prove the main theorem of this section.

Proof of Theorem C.1. Let us condition on the event that Lemma C.3 holds, which happens with probability at least $1 - \Theta(\delta)$. Then, for all $i \in [Q]$, N_i is either at least 0.8, when $B_S(q_j, \vec{r}) \neq \emptyset$, or at most $1 - 0.8 = 0.2$, otherwise. By Lemma B.3, we require that the maximum Laplacian noise not exceed 0.2 with high probability:

$$\Pr[|Z_i| > 0.2] = \Pr\left[|Z_i| > \frac{1}{k}(\ln(1) + 0.2k)\right] \leq e^{-0.2k} \quad (6)$$

Since our threshold for deciding is $\hat{N}_i := N_i + Z_i \geq 0.5$, we can see that setting $k = \Omega(\log(Q/\delta))$ will make the probability in Equation 6 at most $\frac{\delta}{Q}$, implying, by union bound, that \mathcal{A}_{dec} outputs the correct answer at every timestep $i \in [Q]$ with high probability. \square

D PROOF OF THEOREM 1.3

In this section we prove Theorem 1.3 to analyze Algorithm 3. First we show that the vector $\hat{a}(q)$ is produced robustly.

Lemma D.1. *With probability at least $1 - \Theta(\delta)$ over all queries and annuli, the vectors \hat{p} computed by Algorithm 3 are such that:*

$$\|\hat{p} - p\|_\infty \leq 2\eta$$

This holds despite the adversary's action to establish the opposite.

Proof. Our argument mimics the proof of Theorem C.1 in that it invokes the robustification framework of Hassidim et al. (2022).

We first show that producing \hat{p} is $(1, 0)$ -private. Note that it suffices to argue that for a fixed $i \in [k]$ we have $|\hat{p}_i - p_i| \leq \eta$ and \hat{p}_i is produced randomly with respect to the randomness of the copies of

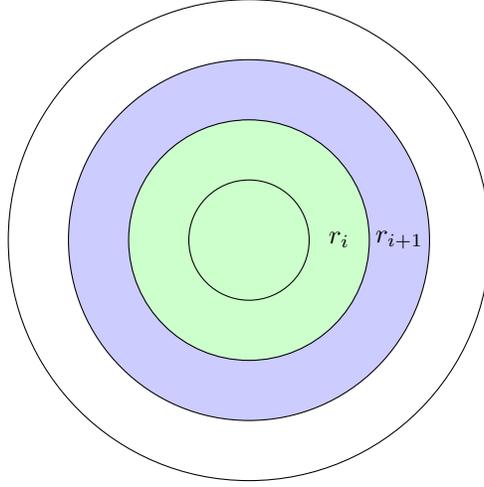


Figure 2: Our concentric LSH construction. In green lies the set $B_S(q, r_i)$, and blue represents the annulus that extends to $B_S(q, r_{i+1})$

$\mathcal{A}_{\text{fair}}$. Indeed, if each \hat{p}_i is produced privately with respect to the input randomness, we can invoke the parallel composition theorem of DP (Theorem B.5) to show that the entire release of \hat{p} is private without additional cost to the privacy parameters.

As we know, privacy with respect to the input randomness implies robustness, so we now have to calculate the cost of privacy in our approximation algorithm.

Fixing some query and $i \in [k]$ we know by the triangle inequality and a standard Chernoff bound (since $m = \Omega(\eta^{-2} \log n)$) that:

$$|p_i - \hat{p}_i| \leq \left| p_i - \frac{1}{s} \sum_{j \in J_i} \tilde{p}_{ij} \right| + \left| \frac{1}{s} \sum_{j \in J_i} \tilde{p}_{ij} - \hat{p}_i \right| \leq \eta + \left| \frac{1}{s} \sum_{j \in J_i} \tilde{p}_{ij} - \hat{p}_i \right|$$

The latter term of the above sum is the error incurred via the privatization process. We can bound it by using our known bound on the magnitude of Laplacian noise (Equation 6):

$$\left| \frac{1}{s} \sum_{j \in J_i} \tilde{p}_{ij} - \hat{p}_i \right| = \left| \frac{1}{s} \sum_{j \in J_i} \tilde{p}_{ij} - \frac{1}{s} \sum_{j \in J_i} \tilde{p}_{ij} + \text{Lap} \left(\frac{1}{s} \right) \right| \leq \frac{1}{s} \eta s = \eta$$

This happens with probability at least $1 - e^{-\eta s} \geq 1 - \frac{1}{\text{poly}(kQ/\delta)}$. Taking a union bound over k annuli and Q queries establishes the lemma. \square

Corollary D.2. *Since Algorithm 3 generates vector $\vec{a} \in \{0, 1\}^k$ by post-processing, this also implies that a is generated robustly.*

Next, we argue that the output point of the algorithm is produced correctly and within the claimed runtime and space complexity.

Theorem D.3. *Algorithm 3 is a $(\delta + \frac{1}{\text{poly}(n)})$ -robust (c, r) -ANN algorithm that uses space $\tilde{O}(\sqrt{Q} \cdot n^{1+\beta})$, where $c' = \sqrt[k]{c}$ and k is an integer chosen to minimize the quantity $\beta = \max\{\rho(c^{1/k}), 1/k\}$. Any query takes $\tilde{O}(dn^\beta)$ time with probability at least 0.998.*

Proof. Letting $\beta = \max\{\rho(c'), 1/k\}$, we maintain $\Theta(k \log^{2.5}(kQn)\sqrt{Q})$ testing instances, as well as k execution instances. This means our total space complexity is:

$$\Theta \left(k \log^{2.5}(kQn/\delta) \sqrt{Q} \cdot n^{1+\rho} + kn^{1+\rho} \right) = \tilde{O}(k\sqrt{Q} \cdot n^{1+\rho}).$$

For the query runtime, suppose $B_S(q, r) \neq \emptyset$. As we argued in Claim 5.2, there must exist some annulus ℓ for which the density ratio is at most $n^{1/k}$. For that annulus, the guarantees of Theorem 3.2

1026 imply that:

$$1027 \Pr \left[T_\ell < 4(n^\rho + n^{1/k}) \log^2(n) \right] \geq 1 - \frac{1}{\text{poly}(n)} \gg 0.999$$

1028 Therefore, there always exists a good annulus when $B_S(q, r) \neq \emptyset$.

1029 By Lemma D.1 we have that a good annulus will, with high probability be captured by Algorithm 3.
 1030 Conversely, if $\hat{a}(q)_i = 1$, then $p_i \geq 0.999 - \eta = 0.998$. As a result, if i^* is the MSB of a ,
 1031 the corresponding execution instance $\mathcal{A}_{\text{fair}}^{(i^*)}$ runs in time $O(dn^\beta)$ with probability at least 0.998.
 1032 Overall, to process one query, we run all k s = $O(\log(Qkn))$ truncated copies $\mathcal{A}_{ij} O(n^{1/k})$. Thus,
 1033 our algorithm takes $O(d \log(nQ) \cdot n^\beta)$ per query, as initially claimed.

1034 Finally, to argue robustness, we know from Lemma D.1 that releasing vector \hat{a} is done robustly.
 1035 Also, Claim 3.3 tells us that the held-out execution copy is robust, given that the MSB i^* is produced
 1036 from a via a fixed function (post-processing). Overall, the output of Algorithm 3 is adversarially
 1037 robust with probability at least $1 - \delta - \frac{1}{\text{poly}(n)}$, accounting for the probability that any of the fair
 1038 ANN algorithms fail. \square

1042 E IMPROVED ROBUST ANNS ALGORITHMS WITH \forall GUARANTEES

1043 In this section, we will discuss another path to adversarial robustness for search problems –providing
 1044 a *for-all* guarantee. We will focus on the ANN problem for this section, due to its ubiquity and
 1045 importance, as well as its amenity to the techniques we discuss.

1048 E.1 A *For-all* GUARANTEE IN THE HAMMING CUBE

1049 We present the Hamming Distance ANN case first because it is the most natural *for-all* guarantee
 1050 one can give. This is because the space we are operating over is discrete, and we can easily union-
 1051 bound over all possible queries and only incur a cost polynomial to the dimension d of the metric
 1052 space.

1053 **Theorem E.1.** *There exists an adversarially robust algorithm solving the (c, r) -ANN problem in the*
 1054 *d -dimensional Hamming Hypercube that can answer every possible query correctly with probability*
 1055 *at least $1 - 1/n^2$. The space requirements are $\tilde{O}(d \cdot n^{1+\rho+o(1)})$, and the time required per query is*
 1056 *$\tilde{O}(d^2 \cdot n^\rho)$, where $\rho = 1/c$.*

1057 *Proof.* First, let us recall the standard LSH in the Hamming Hypercube: We are given a point set
 1058 $S \subseteq \{0, 1\}^d$ with $|S| = n$. We receive queries $q \in \{0, 1\}^d$. Our Locality Sensitive Hash family \mathcal{H}
 1059 is defined as follows: Pick some coordinate $i \in [d]$ and hash $x \in \{0, 1\}^d$ according to $x_i \in \{0, 1\}$.
 1060 This function h acts as a hyperplane separating the points in the hypercube into two equal halves,
 1061 depending on the i -th coordinate. Sampling h uniformly at random from \mathcal{H} is equivalent to sampling
 1062 $i \in [d]$ uniformly at random. We can easily see that \mathcal{H} is an (r, cr, p_1, p_2) -LSH family, as:

$$1063 \Pr_{h \sim \mathcal{H}} [h(p) = h(q)] = \frac{d - \|p - q\|}{d} = \begin{cases} \geq 1 - \frac{r}{d} := p_1, & \text{when } \|p - q\| \leq r \\ \leq 1 - \frac{cr}{d} := p_2, & \text{when } \|p - q\| \geq cr \end{cases}$$

1064 We now go through the typical amplification process for LSH families (Gionis et al., 1999). Instead
 1065 of sampling just one coordinate, we sample k . And instead of sampling just one hash function, we
 1066 sample L different ones $h_1, \dots, h_L \in \mathcal{H}^k$ and require that a close point collides with q at least once.
 1067 With this scheme, we know that if we fix $q \in \{0, 1\}^d$ and $p \in B_S(q, r)$ we have:

$$1068 \Pr [\exists i \in [L] : h_i(p) = h_i(q)] \geq 1 - (1 - p_1^k)^L$$

1069 Furthermore, if $\|p - q\| \geq cr$, we must have:

$$1070 \Pr [\exists i \in [L] : h_i(q) = h_i(p)] \leq Lp_2^k$$

1071 Now, we want to guarantee that with high probability there doesn't exist any query $q \in \{0, 1\}^d$ such
 1072 that for all points $p \in B_S(q, r)$ we have $h_i(q) \neq h_i(p)$ for all $i \in [L]$. In other words, we want:

$$1073 \Pr [\exists q \in \{0, 1\}^d : \forall p \in B_S(q, r) \forall i \in [L] : h_i(p) \neq h_i(q)] \leq \frac{1}{n}$$

We can use the union bound to get:

$$\begin{aligned} & \Pr [\exists q \in \{0, 1\}^d : \forall p \in B_S(q, r) \forall i \in [L] : h_i(p) \neq h_i(q)] \\ & \leq \sum_{q \in \{0, 1\}^d} \Pr [\forall p \in B_S(q, r) \forall i \in [L] : h_i(p) \neq h_i(q)] \end{aligned}$$

So it suffices to establish that for fixed $q \in \{0, 1\}^d$ we have:

$$\Pr [\forall p \in B_S(q, r) \forall i \in [L] : h_i(p) \neq h_i(q)] \leq \frac{1}{n2^d}$$

We can weaken this statement and union-bound as follows:

$$\begin{aligned} \Pr [\forall p \in B_S(q, r) \forall i \in [L] : h_i(p) \neq h_i(q)] & \leq \Pr [\exists p \in B_S(q, r) \exists i \in [L] : h_i(p) = h_i(q)] \\ & \leq \sum_{p \in B_S(q, r)} \Pr [\exists i \in [L] : h_i(p) = h_i(q)] \\ & \leq |B_S(q, r)| \cdot (1 - p_1^k)^L \\ & \leq n(1 - p_1^k)^L \end{aligned}$$

So it suffices to require that:

$$(1 - p_1^k)^L \leq \frac{1}{n^2 2^d} \quad (7)$$

On the other hand, the expected number of points in $S \setminus B_S(q, cr)$ that we will see in the same buckets as q is:

$$\mathbb{E} [|p \in S \setminus B_S(q, cr) \mid \exists i \in [L] : h_i(p) = h_i(q)|] = \sum_{p \in S \setminus B_S(q, cr)} \Pr [\exists i \in [L] \mid h_i(p) = h_i(q)] \quad (8)$$

$$\leq nLp_2^k \quad (9)$$

We can now combine Equation 7 and Equation 9 to work out the values of k and L . First, we want to get $O(L)$ time in expectation, so we require $p_2^k \leq 1/n$, which gives:

$$k \geq \log_{1/p_2}(n)$$

Now, let $p_1 = p_2^\rho$. Substituting, we resolve the value of L as:

$$L \geq n^\rho d \log n$$

With that in place, we can see that our algorithm takes $O(L)$ time with high probability. Indeed, let X be the number of points in $S \setminus B_S(q, cr)$ that are hashed to some common bucket with q . Using a simplified Chernoff bound, we have that:

$$\Pr [X \geq 10L] \leq 2^{-10L} = \frac{1}{n^{10dn^\rho}} \ll \frac{1}{n^{\Omega(1)}}$$

which implies that our runtime per query is $O(L)$ with high probability. As for the value of the constant ρ we have by definition that:

$$\rho := \frac{\log p_1}{\log p_2} = \frac{\log(1 - \frac{r}{d})}{\log(1 - \frac{cr}{d})} \approx \frac{1}{c}$$

Overall, evaluating our hash function requires $\tilde{O}(\log n)$ time, and evaluating distances between points requires $O(d)$ time. We maintain $O(d \cdot n^\rho \log n)$ hash tables, meaning that on a single query we spend $O(d^2 \cdot n^\rho \log n)$ time. For pre-processing, apart from storing the entire dataset in dn space, we take $O(d \cdot n^{1+\rho+o(1)})$ space to construct our data structure. \square

E.1.1 IMPROVING THE QUERY RUNTIME VIA SAMPLING

We can improve the dependency on d for the query runtime by using sampling to find a good bucket. The following theorem encapsulates this finding, reducing the runtime complexity by a factor of d :

Theorem E.2. *There exists an adversarially robust algorithm solving the (c, r) -ANN problem in the d -dimensional Hamming Hypercube that can answer all possible queries correctly with probability at least $1 - 1/n^2$. The space requirements are $\tilde{O}(d \cdot n^{1+\rho+o(1)})$ and the time required per query is $\tilde{O}(d \cdot n^\rho)$, where $\rho = 1/c$.*

1134 *Proof.* From our analysis above, we know that we take $L = n^\rho \cdot d \log n$ different hash functions.
 1135 Consider some query q . We analyze the expected number of buckets that contain some point $p \in$
 1136 $B_S(q, r)$. Let X_q be a random variable representing the number of buckets $i \in [L]$ for which some
 1137 point in $B_S(q, r)$ lies in bucket i . Define the following indicator random variable:

$$1138 \mathbb{1}_i = \begin{cases} 1, & \text{if some point } p \in B_S(q, r) \text{ lies in bucket } i \in [L] \\ 0, & \text{otherwise} \end{cases}$$

1139 By linearity of expectation, we can now write:
 1140

$$1141 \begin{aligned} 1142 \mathbb{E}[X_q] &= \sum_{i=1}^L \Pr[\mathbb{1}_i = 1] \\ 1143 &= \sum_{i=1}^L \Pr \left[\bigcup_{p \in B_S(q, r)} \{h_i(p) = h_i(q)\} \right] \\ 1144 &\geq L \cdot p_1^k \\ 1145 &= L \cdot (p_2)^{\rho k} \\ 1146 &\geq \frac{L}{n^\rho} \\ 1147 &= d \log n \end{aligned}$$

1148 By using the Chernoff bound, we can see that with high probability, X_q is close to its expectation:
 1149

$$1150 \Pr \left[X_q \leq \frac{1}{2} d \log n \right] \leq e^{-\frac{d \log n}{8}} = \frac{1}{n^{d/8}} \ll \frac{1}{n}$$

1151 Let us, then, condition on $X_q > \frac{1}{2} d \log n$. On query time, we can simply sample $m = \Theta(n^\rho \log n)$
 1152 buckets uniformly at random from $[L]$. We know that with probability at least $\frac{d \log n}{2n^\rho d \log n} = \frac{1}{2n^\rho}$, a
 1153 single randomly selected bucket contains some point from $B_S(q, r)$. So, for all m of the selections
 1154 to not contain such a point, the probability is at most:

$$1155 \left(1 - \frac{1}{n^\rho} \right)^{n^\rho \log n} \leq e^{-\log n} = \frac{1}{n}$$

1156 So, with probability at least $1 - \frac{1}{n}$ we find a bucket containing a good point. Since, with high
 1157 probability, the number of points in $P \setminus B_S(q, cr)$ in any bucket are $O(L)$, we see that this sampling
 1158 method improves the query runtime to $O(n^\rho \log n)$. \square

1159 E.1.2 UTILIZING THE OPTIMAL LSH ALGORITHM

1160 Our earlier exposition used the original LSH construction for the Hamming Hypercube (Indyk &
 1161 Motwani, 1998) that achieves $\rho = 1/c$. We can also use the state-of-the-art approach from (Andoni
 1162 & Razenshteyn, 2015) that achieves $\rho = \frac{1}{2c-1}$ in place of Theorem E.1. This slightly improves the
 1163 exponent on n :

1164 **Theorem E.3.** *There exists an adversarially robust algorithm solving the (c, r) -ANN problem in the*
 1165 *d -dimensional Hamming Hypercube that can answer all possible queries correctly with probability*
 1166 *at least 0.99. The space complexity is $O(d \cdot n^{1+\rho+o(1)})$, and the time required per query is $O(d \cdot n^\rho)$,*
 1167 *where $\rho = \frac{1}{2c-1}$. These runtime guarantees hold with high probability.*

1168 The analysis is identical, so we will not repeat it again: Since the algorithm succeeds with constant
 1169 probability, and we want it to succeed on all 2^d possible queries, we boost its success probability
 1170 to $1 - \frac{1}{100 \cdot 2^d}$. This way, after the union bound, any query succeeds with probability at least 0.99.
 1171 Furthermore, the analysis of the sampling algorithm for improving the query runtime in Theorem E.2
 1172 also remains the same. All that changes between using the standard Hamming norm LSH as opposed
 1173 to the optimal one is the ratio $\rho := \frac{\log p_1}{\log p_2}$.

E.2 DISCRETIZATION OF CONTINUOUS SPACES THROUGH METRIC COVERINGS

The *for-all* algorithm we presented as Theorem E.2 cannot be applied outside of discrete spaces, however, because the key to our analysis was the union bound over all the possible queries.

To simulate a similar argument for solving ANN in continuous, ℓ_p spaces, we can consider a strategy of discretizing the space. We place special “marker” points and guarantee that some version of the ANN problem is solvable around them. Then, when a query comes in, we find its corresponding marker point, and solve the ANN problem for it. We show that the answer we get is valid for the original query as well, so long as the “neighborhood” around the marker points is small enough. A similar strategy and covering construction appeared in (Cherapanamjeri & Nelson, 2024), although they did not make algorithmic use of the ability to project any query point to the covering set. Instead, their algorithm deems it sufficient to be successful on every point on just the covering set.

E.2.1 METRIC COVERINGS IN CONTINUOUS SPACES

To initiate our investigation, we need the definition of a *metric covering*:

Definition E.4. Consider a metric space $\mathcal{M} = (\mathbb{R}^d, \|\cdot\|_p)$ with metric μ . Let $U \subset \mathbb{R}^d$ be a bounded subset. A set $\widehat{S} \subseteq \mathbb{R}^d$ is called an Δ -**covering** of U if for all $q \in U$ there exists some $\widehat{s} \in \widehat{S}$ such that

$$\|q - \widehat{s}\|_p \leq \Delta$$

Suppose that U is a bounded subset of \mathbb{R}^d . We can construct the following the following Δ -covering of U : Let $C := \sup_{x \in U} \|x\|_\infty$ and suppose $\{u_i\}_{i=1}^d$ is an orthonormal basis spanning U . We know that $\|x\|_\infty \leq C$ for all $x \in U$, so let us define:

$$\widehat{S} = \sum_{i=1}^d \widehat{\alpha}_i u_i, \quad \text{where}$$

$$\widehat{\alpha}_i \in \{-C, -C + \varepsilon, \dots, C - \varepsilon, C\}$$

for some choice of ε that we will decide later. This is a standard construction for ℓ_2 that we now extend to ℓ_p (Shalev-Shwartz & Ben-David, 2014). As defined, we have:

$$|\widehat{S}| = \left(\frac{2C}{\varepsilon}\right)^d$$

Now, fix some $q \in U$. We can write:

$$q = \sum_{i=1}^d \alpha_i u_i$$

For all $i \in [d]$, let $\widehat{\alpha}_i$ be such that $\alpha_i \in \widehat{\alpha}_i \pm \varepsilon$. Let $\widehat{s} := \sum_{i=1}^d \widehat{\alpha}_i u_i$. Now we have that:

$$\|q - \widehat{s}\|_p^p = \left\| \sum_{i=1}^d (\alpha_i - \widehat{\alpha}_i) u_i \right\|_p^p = \sum_{i=1}^d |\alpha_i - \widehat{\alpha}_i|^p \leq d\varepsilon^p$$

Now, let us set:

$$\varepsilon = \frac{\Delta}{d^{1/p}} \implies \|q - \widehat{s}\|_p \leq \Delta$$

Our construction thus has size:

$$|\widehat{S}| = \left(\frac{2Cd^{1/p}}{\Delta}\right)^d$$

E.2.2 THE ROBUST ANN ALGORITHM

With this construction in mind, our algorithm for robust (c, r) -ANN in ℓ_p space follows as Algorithm 5. The algorithm remains agnostic to the specific LSH data structure that could be used to solve ANN in ℓ_p metric spaces obviously (Charikar, 2002; Datar et al., 2004), but assumes that the success probability over a set of queries in that data structure can be boosted by increasing the number of hash functions taken. This was the case for the Hamming norm as well.

1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295

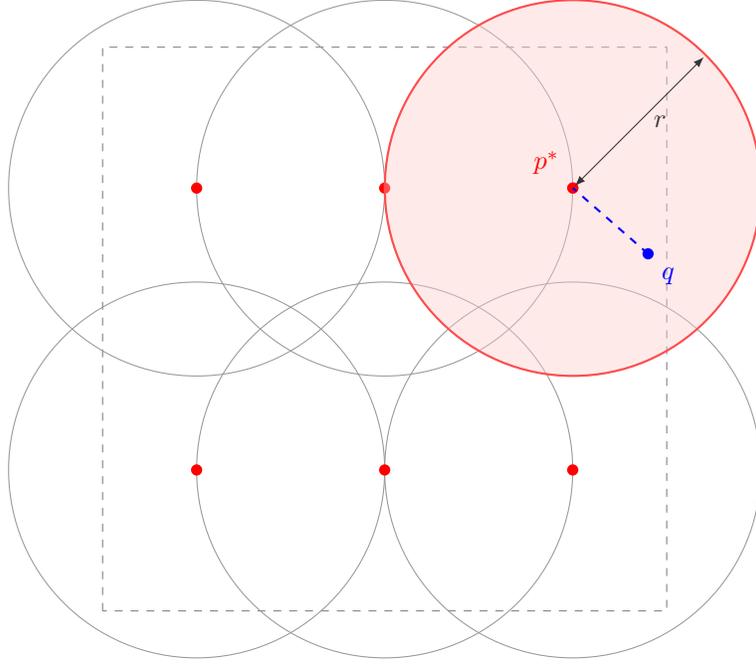


Figure 3: An illustration of an r -covering.

Algorithm 5 Robust ℓ_p ANN through discretization

- 1: *Parameters:* Max-norm C , runtime/accuracy tradeoff $\Delta > 0$, LSH parameters $c, r > 0$.
 - 2: Receive point dataset $S \subset U$ with $|S| = n$ from the adversary.
 - 3: Let \widehat{S} be a Δ -covering of U as constructed in Section E.2.1, and let $c' \leftarrow \frac{cr - \Delta}{r + \Delta}$.
 - 4: Initialize an LSH data structure \mathcal{D} for solving $(c', r + \Delta)$ -ANN that answers all queries in \widehat{S} correctly with high probability.
 - 5: **while** Adversary provides queries **do**
 - 6: Receive query $q \in U$ from the adversary.
 - 7: Find $\widehat{s} \in \widehat{S}$ such that $\|q - \widehat{s}\|_p \leq \Delta$.
 - 8: Query \mathcal{D} on \widehat{s} and output whatever it outputs.
-

Theorem E.5. *There exists an adversarially robust algorithm solving the (c, r) -ANN problem in the (\mathbb{R}^d, ℓ_p) metric space that can answer an unbounded number of adversarial queries. Assuming that the input dataset and the queries are all elements of $U = \{x \in \mathbb{R}^d \mid \|x\|_p \leq C\}$ for some $C > 0$, the pre-processing space is $\widetilde{O}(nT)$ and the time per query is $\widetilde{O}(T)$, where:*

$$T = O \left[d \cdot n^{\rho'} \log \left(\frac{C d^{1/p}}{cr} \right) \right] \quad (10)$$

where:

$$\rho' = \frac{(10 + c)^2}{161c^2 - 20c - 100}$$

Proof. First, to argue for correctness, let q be any query. Suppose there exists some point $x \in S$ with $\|x - q\|_p \leq r$. Then, by triangle inequality it holds that:

$$\|x - \widehat{s}\|_p \leq \|x - q\|_p + \|\widehat{s} - q\|_p \leq \Delta + r$$

Thus, with high probability, \mathcal{D} will find some point $x' \in S$ with $\|x' - \widehat{s}\|_p \leq cr - \Delta$. For that point, we have that:

$$\|x' - q\|_p \leq \|x' - \widehat{s}\|_p + \|\widehat{s} - q\|_p \leq cr - \Delta + \Delta = cr$$

Therefore, Algorithm 5 will output a correct answer. If there doesn't exist such a point x , it is valid for our algorithm to output \perp , so are done.

For the runtime, recall that $|\widehat{S}| \leq O(2Cd^{1/p}/\Delta)^d$. Hence, in order to guarantee success for all queries in \widehat{S} , a similar analysis as to the one for the Hamming Hypercube shows that \mathcal{D} takes up:

$$O \left[d \cdot n^{1+\frac{1}{2c^{j^2}-1}} \log \left(\frac{2Cd^{1/p}}{\Delta} \right) \right]$$

space for pre-processing and

$$O \left[n^{\frac{1}{2c^{j^2}-1}} \log \left(\frac{2Cd^{1/p}}{\Delta} \right) \right]$$

time per query processed, where

$$c' := \frac{cr - \Delta}{r + \Delta}$$

Note that we use the optimal LSH algorithm for ℓ_p spaces, which guarantees $\rho = \frac{1}{2c^2-1}$. Our only constraint is that we must have $\Delta < cr$. If we set $\Delta = \frac{c}{10}r$, we get a per-query runtime of:

$$O \left[n^{1+\frac{1}{2c^{j^2}-1}} \log \left(\frac{20Cd^{1/p}}{cr} \right) \right], \quad \text{where } c' = \frac{9c}{10+c}$$

□

E.2.3 REMOVING THE DEPENDENCY ON THE SCALE

Our algorithm from Theorem E.5 crucially depends on $\log C$, where C is a bounding box for the query and input point space in the ℓ_p norm. We can remove the dependency on C by designing our covering to be data dependent, instead paying an additional logarithmic factor.

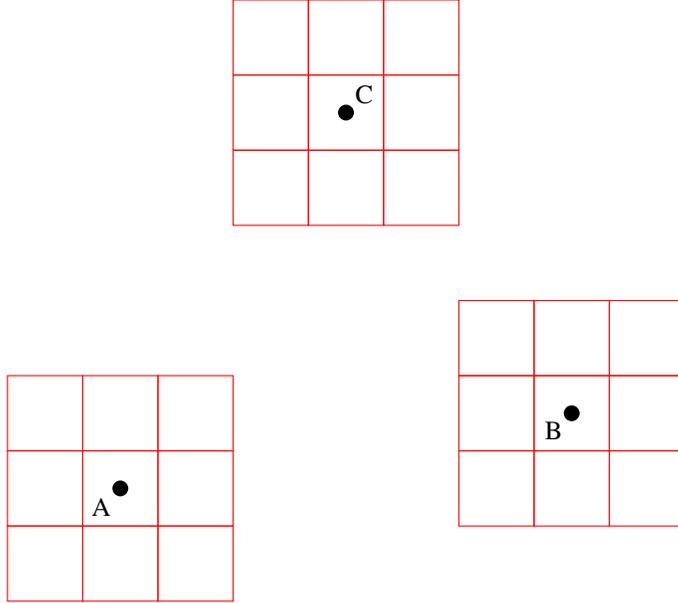


Figure 4: Data-Dependent Discretization of the input query space.

Our new covering \widehat{S}' will be a collection of n Δ -coverings, as constructed in Algorithm 5, each one discretizing the r -ball around a point $p \in S$. The number of points in this new covering is:

$$|\widehat{S}'| \leq O \left[n \cdot \left(\frac{r \cdot d^{1/p}}{cr} \right)^d \right] = O \left[n \cdot \left(\frac{d^{1/p}}{c} \right)^d \right] \quad (11)$$

Note that the size of this covering improves upon the $(nd)^d$ size of the covering given in (Cherapanamjeri & Nelson, 2024), which results in a slightly better runtime. This new covering notably does not cover every possible query. However, it covers exactly the queries we care about. This improved covering leads to the following *for-all* guarantee for robust ANN:

Theorem E.6. *There exists an adversarially robust algorithm solving the (c, r) -ANN problem in the (\mathbb{R}^d, ℓ_p) metric space that can answer an unbounded number of adversarial queries. The pre-processing time / space is $\tilde{O}(nT)$ and the time per query is $\tilde{O}(T/d)$, where:*

$$T = O \left[d \cdot n^{\rho'} (d \log d + \log n) \right] \quad (12)$$

where:

$$\rho' = \frac{1}{2c^2 - 1} = \frac{(10 + c)^2}{161c^2 - 20c - 100}$$

Proof. We distinguish between two cases:

1. If a query q is not included in any $B_S(p, r)$ for any $p \in S$, then the answer can safely be \perp because $B_S(q, r) = \emptyset$ necessarily. Thus, we can just run the default LSH algorithm and simply output whatever it outputs.
2. Otherwise, a query q can be included in some $B_S(p, r)$ for some $p \in S$. Then, suppose $\hat{s}' \in \hat{S}'$ is a point in our covering such that $\|q - \hat{s}'\|_p \leq \Delta$. Then:

$$\|p - \hat{s}'\|_p \leq \|p - q\|_p + \|\hat{s}' - q\|_p \leq r + \Delta \quad (13)$$

Thus, as we argued before, with high probability \mathcal{D} finds some point $x \in S$ with $\|x - \hat{s}'\|_p \leq cr - \Delta$, and for that point we have:

$$\|x - q\|_p \leq \|x - \hat{s}'\|_p + \|\hat{s}' - q\|_p \leq cr - \Delta + \Delta = cr \quad (14)$$

which means our algorithm will output a correct answer.

As before, our algorithm's space and runtime guarantees scale with $\log |\hat{S}'|$. □