# A General Technique for Searching in Implicit Sets via Function Inversion

Boris Aronov* Jean Cardinal† Justin Dallant‡ John Iacono§

**Abstract**

Given a function $f$ from the set $[N]$ to a $d$-dimensional integer grid, we consider data structures that allow efficient orthogonal range searching queries in the image of $f$, without explicitly storing it. We show that, if $f$ is of the form $[N] \to [2^w]^d$ for some $w = \text{polylog}(N)$ and is computable in constant time, then, for any $0 < \alpha < 1$, we can obtain a data structure using $\tilde{O}(N^{1-\alpha/3})$ words of space such that, for a given $d$-dimensional axis-aligned box $B$, we can search for some $x \in [N]$ such that $f(x) \in B$ in time $\tilde{O}(N^{\alpha})$. This result is obtained simply by combining integer range searching with the Fiat-Naor function inversion scheme, which was already used in data-structure problems previously. We further obtain

- data structures for range counting and reporting, predecessor, selection, ranking queries, and combinations thereof, on the set $f([N])$,
- data structures for preimage size and preimage selection queries for a given value of $f$, and
- data structures for selection and ranking queries on geometric quantities computed from tuples of points in $d$-space.

These results unify and generalize previously known results on 3SUM-indexing and string searching, and are widely applicable as a black box to a variety of problems. In particular, we give a data structure for a generalized version of gapped string indexing, and show how to preprocess a set of points on an integer grid in order to efficiently compute (in sublinear time), for points contained in a given axis-aligned box, their Theil-Sen estimator, the $k$th largest area triangle, or the induced hyperplane that is the $k$th furthest from the origin.

## 1 Introduction

The computational problem of *inverting* a function $f$ can be cast as follows: Given a value $y$ in the codomain of $f$, decide if there exists an $x$ such that $f(x) = y$, and return such an $x$ if one exists. The function inversion problem is naturally a fundamental issue in cryptography, and the best known result for general function inversion is due to Fiat and Naor [10]:

THEOREM 1.1. (FIAT-NAOR [10]) *For any function $f\colon [N] \to [N]$, and any choice of a constant $0 < \alpha < 1$, there exists a data structure using $\tilde{O}(N^{1-\alpha/3})$ words of space which can invert $f$ at any given point in $\tilde{O}(N^{\alpha})$ time. Moreover, this data structure can be built in $\tilde{O}(N)$ time with high probability.*

(Note that we use $[N]$ to denote $\{0, \ldots, N-1\}$, and use the asymptotic $\tilde{O}$ notation, that omits polylogarithmic factors.)

As was noticed previously by different sets of authors [8, 13, 17], Theorem 1.1 can be generalized to the case where the codomain of $f$ might be larger than the domain of $f$, by using $O(\log N)$ hash functions from a family of pairwise independent functions.

COROLLARY 1.1. ([8, 13, 17]) *Theorem 1.1 holds for any function $f\colon [N] \to [2^w]$, with $w = \text{polylog}(N)$.*

Recently, the above data structures for function inversion have been applied to tackle fundamental problems that do not directly stem from cryptography:

*Department of Computer Science and Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA; boris.aronov@nyu.edu. Work partially supported by NSF Grant CCF-20-08551.

†Université libre de Bruxelles (ULB), CP212, Boulevard du Triomphe, 1050 Brussels, Belgium; jean.cardinal@ulb.be.

‡Université libre de Bruxelles (ULB), CP212, Boulevard du Triomphe, 1050 Brussels, Belgium; justin.dallant@ulb.be. Supported by the French Community of Belgium via the funding of a FRIA grant.

§Université libre de Bruxelles (ULB), CP212, Boulevard du Triomphe, 1050 Brussels, Belgium; john.iacono@ulb.be. This work was supported by the Fonds de la Recherche Scientifique-FNRS under Grant n° 40008322.

**3SUM-indexing** [17, 13]. Preprocess two sets $S_1$ and $S_2$ of $n$ integers each, so that for any integer $y$, one can decide whether there exist $x_1 \in S_1$ and $x_2 \in S_2$ such that $y = x_1 + x_2$.

**Collinearity indexing with queries on a line** [1]. Preprocess two sets $S_1$ and $S_2$ of $n$ points in the plane, together with a line $\ell$, so that for any point $y \in \ell$, one can decide whether there exist $p_1 \in S_1$ and $p_2 \in S_2$ such that $y$, $p_1$, and $p_2$ are collinear.

**Gapped string searching** [4]. Preprocess a string of length $n$ so that given two patterns $P_1$ and $P_2$ and an integer interval $[\delta, \beta]$, one can report all joint occurrences of $P_1$ and $P_2$ separated by a distance in the interval $[\delta, \beta]$.

It is a striking fact that, for each of these problems, the best known space-time trade-offs are achieved using such a general tool as the Fiat-Naor inversion scheme. The purpose of this contribution is to extend the applicability of this tool. We show that function inversion, used as a black box, allows to not only decide whether there exists some $x$ such that $f(x) = y$, but to actually search, select, and rank values in the image of $f$, as well as to restrict those values to lie in (axis-aligned) boxes[1] given as query arguments. The space-time tradeoffs are essentially the same as those achieved for function inversion: $k$-ary queries on $n$ items can be answered in strongly sublinear time from a data structure using $O(n^{k-\alpha})$ space, for some $\alpha > 0$. Our contribution can therefore be seen as an efficient reduction of various problems to the function inversion problem. This provides a simple and powerful method to design data structures for *implicit set representations*, as defined by Chazelle [7].

The main idea underlying all our results is that the elementary operations used in range searching, essentially navigating a hierarchy of dyadic boxes, can themselves be implemented by inverting another function, with a slightly larger domain and codomain but within essentially the same space and time bounds. The algorithms are elementary, provided that the function inversion data structure is available as a black box. It can be implemented using any such data structure, including some that would be suboptimal but easier to implement. Conversely, any improvement on the function inversion data structure translates mechanically to an improved algorithm.

In the next section, we consider range searching and range counting queries in the image of an integer function. In Section 3, we use these two basic results to provide a collection of data structures that allow to answer complex queries involving predecessor, ranking, and selection queries in boxes, both in the image and domain of an integer function. In Section 4, we illustrate the versatility of the framework on a number of elementary problems. These include generalizations of $k$-SUM indexing data structures and string searching, but also new geometric data structures.

## 2 Range queries

We state our two main technical results, involving data structures for range searching and range counting queries in the image of a $d$-dimensional function $f$.

THEOREM 2.1. (RANGE SEARCHING) *Let $f\colon [N] \to [2^w]^d$ be a function computable in $\tilde{O}(1)$ time, where $w = \mathrm{polylog}(N)$ and $d = O(1)$. For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(N)$ time with high probability a data structure using $\tilde{O}(N^{1-\alpha/3})$ words of space which supports the following query: Given a $d$-dimensional box $B$, test if there exists $x \in [N]$ with $f(x) \in B$, and report such a value if it exists, in $\tilde{O}(N^{\alpha})$ time.*

*Proof.* Let $g\colon [N] \times [w+1]^d \to [2^{w+1}]^d \times [w+1]^d$ be the function defined as follows:

$$g(x, i_1, i_2, \ldots, i_d) \coloneqq \left( \left\lfloor \frac{f_1(x)}{2^{i_1}} \right\rfloor, \left\lfloor \frac{f_2(x)}{2^{i_2}} \right\rfloor, \ldots, \left\lfloor \frac{f_d(x)}{2^{i_d}} \right\rfloor, i_1, i_2, \ldots, i_d \right).$$

The domain of this function has size $O(N(w+1)^d) = \tilde{O}(N)$ and its codomain has size $O(2^{d(w+1)}(w+1)^d) = \tilde{O}(2^{\mathrm{polylog}(N)})$, so we can apply Corollary 1.1 to obtain a data structure using $\tilde{O}(N^{1-\alpha/3})$ words of space which can invert $g$ at any given point in $\tilde{O}(N^{\alpha})$ time.

[1] Throughout the paper, we use the term "box" as short for "axis-aligned box", i.e. a hyperrectangle whose sides are parallel to the coordinate axes.

We call a $d$-dimensional box *dyadic* if the coordinates defining the limits of the box in each dimension are successive multiples of a power of two. We can specify such a box using the multiples and powers of two in each dimension:

$$DB(y,i) = DB(y_1, y_2 \ldots, y_d, i_1, i_2, \ldots i_d) \coloneqq \\ [2^{i_1}y_1, 2^{i_1}(y_1+1)-1] \times [2^{i_2}y_2, 2^{i_2}(y_2+1)-1] \times \ldots \times [2^{i_d}y_d, 2^{i_d}(y_d+1)-1].$$

Our function $g$ has been defined so that

$$f(x) \in DB(y,i) \longleftrightarrow g(x,i) = (y,i).$$

Thus, given any dyadic box $DB(y,i)$, we can determine if there is an $x$ such that $f(x) \in DB(y,i)$ by asking if $(y,i)$ is in the image of $g$.

Given a dyadic box $DB(y,i)$ that contains $f(x)$ for some $x$ that we wish to compute, each dyadic box with $d' \leq d$ non-zero elements of $i$ can be partitioned into $2^{d'}$ dyadic boxes where the non-zero elements of $i$ are decremented by one; call these the child boxes of $DB(y,i)$. These child boxes can equivalently be viewed as the result of cutting $DB(y,i)$ in half in each dimension where its size is at least 2. If some $f(x) \in DB(y,i)$, then $f(x) \in B'$ for some child box $B'$ of $DB(y,i)$; continue this process until some $f(x) \in DB(z,0)$ is found, which means $f(x) = z \in DB(y,i)$. This will involve $w+1$ steps, each consisting of at most $2^d$ inversion queries.

Let $B$ be a $d$-dimensional box, not necessarily dyadic. To determine if there is an $f(x) \in B$, we use the fact that any $d$-dimensional box defined by integers at most $2^{\text{polylog}(N)}$ can be partitioned into $O(\log^d(2^{\text{polylog}(N)})) = \tilde{O}(1)$ dyadic boxes, and query each of these separately. ∎

Note that in one dimension the idea of searching in bounded universes via hashing is well known, from, for example, the $x$- and $y$-fast tries of Willard [19]. The idea of extending this to higher dimensions with dyadic boxes has been used in the past, for example in the point location method of Iacono and Langerman [15].

We now show how to extend our result from searching to counting using a big/small approach whereby counts of dyadic boxes intersecting large number of images of $f$ have their counts stored explicitly, and the remaining dyadic boxes have their counts computed on demand using function inversion. This increases the space usage from $\tilde{O}(N^{1-\alpha/3})$ to $\tilde{O}(N^{1-\alpha/4})$ while supporting $\tilde{O}(N^\alpha)$-time queries.

THEOREM 2.2. (RANGE COUNTING) *Let $f\colon [N] \to [2^w]^d$ and $c\colon [2^w]^d \to [2^w]$ be functions computable in $\tilde{O}(1)$ time, where $w = \text{polylog}(N)$ and $d = O(1)$. For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(N)$ time with high probability a data structure using $\tilde{O}(N^{1-\alpha/4})$ words of space which supports the following query: Given a $d$-dimensional box $B$, return $\sum_{y \in f([N]) \cap B} c(y)$, in $\tilde{O}(N^\alpha)$ time.*

*Proof.* Consider again the function $g$ defined in the proof of Theorem 2.1, with the same inversion data structure using $\tilde{O}(N^{1-\alpha/3})$ words of space which can invert $g$ at any given point in $\tilde{O}(N^\alpha)$ time.

For every dyadic box $DB(y,i)$ which contains at least $N^{\alpha/3}$ elements of $f([N])$, store that box as a key with value $\sum_{y \in f(N) \cap DB(y,i)} c(y)$ in a dictionary (which can be implemented in any number of ways, for instance using hash tables or binary search trees). As each element of $f([N])$ appears in at most $O(\log^d(2^{\text{polylog}(N)})) = \tilde{O}(1)$ dyadic boxes, the number of dyadic boxes with at least $N^{\alpha/3}$ elements is at most $\tilde{O}(N^{1-\alpha/3})$. Thus, this does not increase space usage significantly.

Now, suppose we are given a dyadic box $DB(y,i)$ and want to find the number of elements of $f([N])$ in that box. We start by checking if this box contains at least $N^{\alpha/3}$ elements by a lookup in the dictionary, and we can report the stored number if it is the case. Otherwise, the box contains at most $N^{\alpha/3}$ elements and we recurse on all non-empty children of $DB(y,i)$. We keep a global counter $s$ to which we add $c(z)$ each time we reach non-empty box of the form $DB(z,0)$. Because each element of $f([N])$ appears in at most $\tilde{O}(1)$ dyadic boxes, this requires $\tilde{O}(N^{\alpha/3})$ inversion queries, for a total query time of $\tilde{O}(N^{\alpha/3} \cdot N^\alpha) = \tilde{O}(N^{4\alpha/3})$.

Given a non-dyadic box, we can again partition it into $O(\log^d(2^{\text{polylog}(N)})) = \tilde{O}(1)$ dyadic boxes and query each of these separately.

Finally, re-scaling $\alpha$ by a factor of 3/4 gives the result. ∎

The previous proof can easily be adapted to support other types of associative and commutative operations instead of a sum, such as taking the maximum in a box.

## 3 Beyond range queries

We now give general results on the existence of data structures for distinct or more complex types of queries. The proofs are elementary and only rely on Theorems 2.1 and 2.2. Note that some of these results can be proved directly using similar ideas as for Theorems 2.1 and 2.2 and with a slightly better running time if we take polylogarithmic factors into account.

**3.1 Predecessor, ranking, and selection queries** We first observe that the range searching data structures described above directly yield data structures for predecessor, ranking, and selection queries.

In the following two lemmas, we let $f\colon [N] \to [2^w]$ be a function computable in $\tilde{O}(1)$ time, where $w = \text{polylog}(N)$.

THEOREM 3.1. (PREDECESSOR SEARCH) *For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(N)$ time with high probability a data structure using $\tilde{O}(N^{1-\alpha/3})$ words of space which supports the following query: Given an integer $y \in [2^w]$, report $x \in [N]$ such that $f(x)$ is the largest value in $f([N])$ that is smaller or equal to $y$, if it exists, in $\tilde{O}(N^\alpha)$ time.*

*Proof.* We bisect on the integers $z \in [y]$. For such an integer $z$, we define $B \coloneqq [z, y-1]$ and use the data structure of Theorem 2.1 with $d = 1$ to decide whether there exists $x$ such that $f(x) \in B$ in time $\tilde{O}(N^\alpha)$. The number of such queries is at most $\log(2^w) = w = O(\text{polylog}(N))$, hence the overall cost remains $\tilde{O}(N^\alpha)$. ∎

THEOREM 3.2. (RANKING AND SELECTION) *For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(N)$ time with high probability a data structure using $\tilde{O}(N^{1-\alpha/4})$ words of space which supports the following queries in $\tilde{O}(N^\alpha)$ time:*

**Ranking.** *Given an integer $y \in [2^w]$, return $\big|\{x \in [N] : f(x) < y\}\big|$.*

**Selection.** *Given an integer $k \in [N]$, report $x \in [N]$ such that $f(x)$ is the $k$th largest value in $f([N])$, if it exists.*

*Proof.* For ranking, we let the function $c$ be such that $c(x) = 1$ for all $x \in [2^w]$, and use the data structure of Theorem 2.2 with $d = 1$. Querying the interval $B \coloneqq [y]$ in this structure yields the answer in time $\tilde{O}(N^\alpha)$.

For selection, we bisect over the values $y \in [2^w]$ and repeatedly query the data structure on the interval $B \coloneqq [y]$. This requires $O(\text{polylog}(N))$ steps, hence time $\tilde{O}(N^\alpha)$ overall as well. ∎

**3.2 Ranking and selection with range constraints** By applying Theorems 2.1 and 2.2 with $d > 1$, we can also perform predecessor, ranking, and selection queries within a given box.

THEOREM 3.3. *Let $f\colon [N] \to [2^w]^d$ and $\mu\colon [2^w]^d \to [2^w]$ be functions computable in $\tilde{O}(1)$ time, where $w = \text{polylog}(N)$ and $d = O(1)$. For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(N)$ time with high probability a data structure using $\tilde{O}(N^{1-\alpha/4})$ words of space which supports the following queries in $\tilde{O}(N^\alpha)$ time:*

**Ranking in a box.** *Given a $d$-dimensional box $B$ and an integer $y \in [2^w]$, return the value*
$\big|\{f(x) : x \in [N], f(x) \in B, \mu(f(x)) < y\}\big|$.

**Selection in a box.** *Given a $d$-dimensional box $B$ and an integer $k \in [N]$, report $x \in [N]$ such that $\mu(f(x))$ is the $k$th largest value in $\{\mu(f(x)) : x \in [N], f(x) \in B\}$, if it exists.*

**Median in a box.** *Given a $d$-dimensional axis aligned box $B$, report $x \in [N]$ such that $\mu(f(x))$ is the median of $\{\mu(f(x)) : x \in [N], f(x) \in B\}$.*

*Proof.* To perform ranking of $y$ in a box $B$, we can construct the data structure of Theorem 2.2 on the function $g\colon [N] \to [2^w]^{d+1}$ defined by

$$g(x) \coloneqq (f(x), \mu(f(x))),$$

let the count function $c$ for the data structure be equal to 1 everywhere, and query the data structure with the $(d+1)$-dimensional box $B \times [y]$ in time $\tilde{O}(N^\alpha)$.

Selection in a box can be done as for ranking, except we now bisect on the values $y$, which only increases the running time by a $O(\text{polylog}(N))$ factor.

Finally, we can select the median by first counting the number $m$ of values in the preimage of $B$, then performing selection with $k = m/2$. ∎

Note that this result can be used to select a value among preimages, as follows.

THEOREM 3.4. (PREIMAGE QUERIES) *Let $f\colon [N] \to [2^w]$ be a function computable in $\tilde{O}(1)$ time, where $w = \text{polylog}(N)$. For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(N)$ time with high probability a data structure using $\tilde{O}(N^{1-\alpha/4})$ words of space which supports the following queries in $\tilde{O}(N^\alpha)$ time:*

**Preimage ranking.** *Given two integers $y, z \in [2^w]$, return $\big|\{x \in [N] : f(x) = y, x < z\}\big|$.*

**Preimage selection.** *Given an integer $k \in [N]$ and an integer $y \in [2^w]$, report the $k$th largest value $x \in [N]$ such that $f(x) = y$, if it exists.*

**Preimage median.** *Given an integer $y \in [2^w]$, return the median of the values $x \in [N]$ such that $f(x) = y$, if such a value exists.*

*Proof.* Let $g\colon [N] \to [2^w] \times [N]$ be the function defined by

$$g(x) \coloneqq (f(x), x).$$

Preimage ranking, selection, and median then reduces directly to ranking, selection, and median in a box respectively, on the function $g$. The result then follows from Theorem 3.3. ∎

**3.3 Range reporting** One can also consider the problem of *reporting* distinct values $x$ such that $f(x)$ is contained in a specific box. The following can be obtained from Theorem 2.1.

THEOREM 3.5. (RANGE REPORTING) *Let $f\colon [N] \to [2^w]^d$ be computable in $\tilde{O}(1)$ time, where $w = \text{polylog}(N)$ and $d = O(1)$. For any choice of $0 < \alpha < 1$, we can construct in $\tilde{O}(N)$ time with high probability a data structure using $\tilde{O}(N^{1-\alpha/3})$ words of space which supports the following query in $\tilde{O}((k+1)N^\alpha)$ time: Given a $d$-dimensional box $B$ and and an integer $k \in [N]$, return $k$ distinct values $x \in [N]$ such that $f(x) \in B$, if they exist.*

*Proof.* From the data structure of Theorem 2.1 constructed on the function $g(x) \coloneqq (f(x), x)$, we can decide in time $\tilde{O}(N^\alpha)$ whether there exists any $x \in [N]$ such that $f(x) \in B$, and $x < y$ for a given box $B$ and integer $y$. This allows to perform a bisection search on the coordinates of $B$ and $y$ and isolate $k$ distinct values $x$ such that $f(x) \in B$, in $O(\text{polylog}(N))$ time for each. ∎

## 4 Applications

We can apply our general method not only to new variants of problems for which function inversion was already known to be useful [12, 13, 17, 4, 1], but also to indexing variants of a number of other previously studied problems in computational geometry [7, 3, 2, 6].

**4.1 Variants of 3SUM-indexing** The 3SUM-indexing problem consists of preprocessing two sets $S_1$ and $S_2$, each containing $n$ integers, so that for any integer $y$, one can quickly decide whether there exist $x_1 \in S_1$ and $x_2 \in S_2$ such that $y = x_1 + x_2$. It is therefore a data structure version of the classical 3SUM problem [11, 14, 16]. Achievable space-time tradeoffs for this question were the topic of several recent results. It was conjectured by Goldstein, Kopelowitz, Lewenstein, and Porat [12] that any data structure for 3SUM-indexing with $O(n^\alpha)$ query time for some $\alpha < 1$ must use $\Omega(n^2)$ space. This conjecture was disproved by Golovnev, Guo, Horel, Park, and Vaikuntanathan [13], and Kopelowitz and Porat [17] using Corollary 1.1. In fact, the 3SUM-indexing problem is a prominent example of a problem that is currently best solved by using this machinery. This is surprising, considering the fact that it does not rely on any algebraic property of the problem.

As an example of a simple generalization that we can tackle using our results, we consider the following generalization of the $k$-SUM-indexing problem to arbitrary constant-degree polynomial functions, yielding a data structure version of the recently studied 3POL and $k$-POL problems [2, 6].

PROBLEM 1. ($k$-POL-INDEXING PROBLEM) *Let $S_1, S_2, \ldots, S_k$ be $k$ sets of $n$ numbers in $[2^w]$, where $k = O(1)$ and $w = O(\text{polylog}(n))$, and let $p \in \mathbb{N}[x_1, x_2, \ldots, x_k]$ be a constant-degree $k$-variate polynomial with coefficients in $[2^w]$. Preprocess these sets and $p$ to answer queries of the following form: given $k$ intervals $B_1, B_2, \ldots, B_k \subset [2^w]$ and an integer $y$, are there $k$ values $x_1 \in B_1 \cap S_1, x_2 \in B_2 \cap S_2, \ldots, x_k \in B_k \cap S_k$ such that $p(x_1, x_2, \ldots, x_k) = y$?*

THEOREM 4.1. ($k$-POL-INDEXING PROBLEM) *For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(n^k)$ time with high probability a data structure for the $k$-POL-indexing problem using $\tilde{O}(n^{k-\alpha/3})$ words of space which supports queries in $\tilde{O}(n^\alpha)$ time.*

*Proof.* We can assume without loss of generality that the values of the polynomial $p$ lie in $[2^{w'}]$ for some $w' \geq w$ such that $w' = O(\text{polylog}(n))$. We define the function $f\colon [n^k] \to [2^{w'}]^{k+1}$ that maps a $k$-tuple $(i_1, i_2, \ldots, i_k)$ to the corresponding $(k+1)$-tuple of integers $(x_1, x_2, \ldots, x_k, p(x_1, x_2, \ldots, x_k))$, where $x_j$ is the $i_j$th largest value in $S_j$. We then construct the data structure of Theorem 2.1 with $d = k+1$ and $N = n^k$, and use it to answer queries with the $(k+1)$-dimensional boxes $B \coloneqq B_1 \times B_2 \times \ldots \times B_k \times \{y\}$. ∎

Note that from Theorem 3.3, with a slight penalty in space, we can also answer ranking and selection queries in the preimage $p^{-1}(y) \cap B$ in sublinear time.

**4.2 Substring search** Function inversion has been applied to string searching problems before [8, 4]. Let us first consider the following problem:

PROBLEM 2. (BIRANGE PROXIMITY) *Given an array $A$ of size $n$ containing elements of $[n]$. Preprocess it to answer queries of the form: given intervals $[i, j]$, $[k, l]$, and $[\delta, \beta]$ all with endpoints in $[n]$, report all pairs $(x, y)$, $x \in [i, j]$ and $y \in [k, l]$, such that $|A[x] - A[y]| \in [\delta, \beta]$.*

We can define $f\colon [n]^2 \to [n]^3$ by $f(x, y) \coloneqq (x, y, |A[x] - A[y]|)$. Answering a birange proximity query is thus equivalent to determining the pairs $(x, y)$ with $f(x, y) = (x, y, \gamma)$, $x \in [i, j]$, $y \in [k, \ell]$, and $\gamma \in [\delta, \beta]$. Thus, by applying Theorem 3.5 we immediately obtain:

LEMMA 4.1. (BIRANGE PROXIMITY) *For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(n^2)$ time with high probability a data structure for the birange proximity problem using $\tilde{O}(n^{2-\alpha/3})$ words of space which supports queries that return $k$ pairs in time $\tilde{O}((k+1)n^\alpha)$.*

Bille et al. [4] consider the following problem:

PROBLEM 3. (GAPPED STRING INDEXING) *Given a string $S$ of length $n$, prepossess it to support queries of the form: Given two strings $P_1$ and $P_2$, and an interval $[\delta, \beta]$, report all $(x, y)$ such that $P_1$ is a substring starting at location $x$ in $S$, $P_2$ is a substring starting at location $y$ in $S$, and $|x - y| \in [\delta, \beta]$.*

The suffix array $A$ of a string $S$ is a fundamental data structure in stringology due to Manber and Meyers [18]. It has $A[j] = i$ if $S[i \ldots n]$ is the $j$th lexicographically largest suffix of $S$, and it can be computed in linear time using the algorithm of Farach-Colton [9]. Bille et al. [4] observe that given the suffix array $A$ of $S$, which is a permutation of $[n]$, the locations of of occurrences of a substring $P$ are the values in a consecutive interval of $A$, and this interval can be computed in time $O(|P|)$ using a suffix tree (which can also be computed in linear time). Using a birange proximity structure, initialized with the suffix array of $S$, a gapped string indexing query in $S$ can be converted into a birange proximity query at cost $O(|P_1| + |P_2|)$. We therefore immediately recover the headline result of [4]:

THEOREM 4.2. (GAPPED STRING INDEXING) *For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(n^2)$ time with high probability a data structure for the gapped string indexing problem using $\tilde{O}(n^{2-\alpha/3})$ words of space which supports queries that return $k$ pairs in time $\tilde{O}(|P_1| + |P_2| + (k+1)n^\alpha)$.*

While the problem we call birange proximity is implicit in the presentation of Bille et al. [4], their reductions make use of other problems; here the use of birange proximity along with the framework we have developed makes the proof immediate and amenable to many possible variations. We state one possible generalization, where instead of two strings, $k$ substrings are to be matched, with $k \geq 2$ a constant. We impose pairwise restrictions on the allowable distances between the matches, and additionally we restrict the occurrence of each substring to an interval in the base string.

PROBLEM 4. (GENERALIZED GAPPED STRING INDEXING) *Given a string $S$ of length $n$ and a constant nonnegative integer $d$, preprocess to support queries of the form: Given strings $P_0, P_1, \ldots P_{d-1}$, intervals $[\delta_i, \beta_j]$ for all $i, j \in [d]$, $i < j$, intervals $[\gamma_i, \xi_i]$, for all $i \in [d]$ report all $(x_0, x_1, \ldots x_{d-1})$ such that $P_i$ is a substring starting at location $x_i$ in $S$, $x_i \in [\gamma_i, \xi_i]$ and $|x_i - x_j| \in [\delta_i, \beta_j]$ for all $i, j \in [d]$, $i < j$. The counting variant returns the number of $(x_0, x_1, \ldots, x_{d-1})$ meeting this condition.*

Using Theorems 3.3 and 3.5 with $N = \tilde{O}(n^k)$, we directly obtain the following.

THEOREM 4.3. (GENERALIZED GAPPED STRING INDEXING) *For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(n^d)$ time with high probability a data structure for the generalized gapped string indexing problem using $\tilde{O}(n^{d-\alpha/3})$ words of space which supports queries that return $k$ pairs in time $\tilde{O}(\sum_{i=1}^{d} |P_i| + (k+1)n^\alpha)$. For the counting variant, queries take time $\tilde{O}(\sum_{i=1}^{d} |P_i| + n^\alpha)$ and the space usage is $\tilde{O}(n^{d-\alpha/4})$.*

**4.3 Geometric applications** The following result follows directly from Theorem 3.3 and will be useful in applying our method to specific computational geometry problems.

LEMMA 4.2. (SEARCHING TUPLES OF POINTS IN SPACE) *Let $S$ be a set of $n$ points on the $d$-dimensional integer grid $[2^w]^d$, where $w = \text{polylog}(n)$ and $d = O(1)$. Let $\delta : [2^w]^{td} \to [2^w]$, for some constant $t > 1$, be any integer function mapping a $t$-tuple of points in $S$ to an integer, and computable in $\tilde{O}(1)$.*

*For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(n^t)$ time with high probability a data structure using $\tilde{O}(n^{t-\alpha/4})$ words of space which supports the following queries in $\tilde{O}(n^\alpha)$ time:*

**Ranking in a box.** *Given $t$ $d$-dimensional boxes $B_1, B_2, \ldots, B_t$ and an integer $y$, return the number of $t$-tuples $T$ of distinct points of $S$ in $B_1 \times B_2 \times \cdots \times B_t$ such that $\delta(T) < y$.*

**Selection in a box.** *Given $t$ $d$-dimensional boxes $B_1, B_2, \ldots, B_t$ and an integer $k$, return the $k$th largest value of $\delta(T)$ among all $t$-tuples $T$ of distinct points of $S$ in $B_1 \times B_2 \times \cdots \times B_t$.*

**Median in a box** *Given $t$ $d$-dimensional boxes $B_1, B_2, \ldots, B_t$, return the median of $\delta(T)$ among all $t$-tuples $T$ of distinct points of $S$ in $B_1 \times B_2 \times \cdots \times B_t$.*

*Proof.* We denote by $p_{i,\ell}$ the $\ell$th coordinate of the $i$th point of $S$, for $i \in [n]$. Consider the function $g$ mapping an ordered $t$-tuple of pairwise distinct elements of $[n]$ to $[2^w]^{td}$, defined as

$$g((i_1, i_2, \ldots, i_t)) \coloneqq (p_{i_1,1}, p_{i_1,2}, \ldots, p_{i_1,d}, p_{i_2,1}, p_{i_2,2}, \ldots, p_{i_2,d}, \ldots, p_{i_t,1}, p_{i_t,2}, \ldots, p_{i_t,d}).$$

The result follows by applying Theorem 3.3 to the functions $g$ and $\delta$, with $N = n!/(n-t)! < n^t$. ∎

One simple example of a function $\delta$ for $t = 2$ is the squared Euclidean distance between two points, and we then get a data structure for interdistance selection and ranking. For $t = 3$, we can also define the function $\delta$ as twice the area of the triangle defined by the three points. Note that from Pick's Theorem, the area is half-integer. Lemma 4.2 then directly yields a data structure for the following problem.

PROBLEM 5. (TRIANGLE SELECTION IN BOXES) *Given a set $S$ of $n$ points on the two-dimensional integer grid $[2^w]^2$, where $w = \text{polylog}(n)$, preprocess it to answer queries of the form: given three two-dimensional boxes $B_1, B_2, B_3$ and an integer $k$, return the triangle with the $k$th largest area among all those formed by three points in $(S \cap B_1) \times (S \cap B_2) \times (S \cap B_3)$.*

THEOREM 4.4. (TRIANGLE SELECTION IN BOXES) *For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(n^3)$ time with high probability a data structure for the triangle selection in boxes problem using $\tilde{O}(n^{3-\alpha/4})$ words of space which supports queries in $\tilde{O}(n^\alpha)$ time.*

This problem, as well as the interdistance selection problem, have previously been considered by Chazelle [7] in the "one-shot" setting where there are no query boxes, $k$ is fixed, and the points have real coordinates.

Other problems of interest involve functions with divisions or radicals, for which our approach focusing on integer functions does not seem to apply at first glance. In many cases we can circumvent this problem, due to the fact that a limited precision is enough to decide which of two values is larger. One bound implied for example by the work of Burnikel et al. [5] is the following.

LEMMA 4.3. ([5]) *Let $f : [2^w]^d \to \mathbb{R}$ be a function which has a constant-sized expression consisting of the operators $+, -, \times, /, \sqrt{\cdot}$, $O(w)$-bit constants and the arguments of $f$. Then computing $f$ up to $O(w)$ bits of precision is enough to decide if $f(p) \leq f(q)$ for any $p, q \in [2^w]^d$.*

In what follows, given a function $f$ for which this lemma applies, we will use the notation $f^\bullet$ to denote the function computed to the required $O(w)$ of bits of precision, scaled to have an integer domain. Thus, $f^\bullet$ is a function $[2^w]^d \to [2^{O(w)}]$. Note that $f^\bullet$ is always computable in $O(\text{poly}(w)) = \tilde{O}(1)$ time.

An example here is the case $t = 2$, $d = 2$, and $\delta(p_1, p_2) = f^\bullet(p_1, p_2)$, where $f(p_1, p_2)$ is defined as the slope of the line through $p_1, p_2$. In robust statistics, the *Theil-Sen estimator* of a set $S$ of points in $\mathbb{R}^2$ is the median of the slopes of all the lines through two points of $S$. It is considered as a more robust estimator for linear regression than the classical least-square estimator, and commonly used in practice.

PROBLEM 6. (LINEAR REGRESSION IN A BOX) *Given a set $S$ of $n$ points on the two-dimensional integer grid $[2^w]^2$, where $w = \text{polylog}(n)$, preprocess it to answer queries of the form: given a two-dimensional box $B$, return the Theil-Sen estimator of the points in $S \cap B$.*

Applying Lemma 4.2, we directly obtain the following.

THEOREM 4.5. (LINEAR REGRESSION IN A BOX) *For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(n^2)$ time with high probability a data structure for linear regression in a box using $\tilde{O}(n^{2-\alpha/4})$ words of space which supports queries in $\tilde{O}(n^\alpha)$ time.*

Another example is the following problem, which is a data structure variant of a problem previously considered by Bespamyatnikh and Segal [3].

PROBLEM 7. (HYPERPLANE DISTANCE SELECTION) *Given $n$ points on the $d$-dimensional grid $[2^w]^d$, preprocess them to answer queries of the following form: given $k$ and $d$ boxes $B_1, \ldots, B_d$, return the hyperplane with the $k$th largest distance to the origin, among all hyperplanes spanned by $d$ points in $B_1 \times B_2 \times \cdots \times B_d$.*

Let $f$ be the function which maps $d$-tuples of points in $[2^w]^d$ to the distance between the origin and the hyperplane they span. Note that for any constant $d$, we can express this function in a way to which Lemma 4.3 applies. Thus, applying Lemma 4.2 with $\delta = f^\bullet$ yields the following.

THEOREM 4.6. (HYPERPLANE DISTANCE SELECTION) *For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(n^d)$ time with high probability a data structure for hyperplane distance selection using $\tilde{O}(n^{d-\alpha/4})$ words of space which supports queries in $\tilde{O}(n^\alpha)$ time.*

As a last example, let us consider the following problem, which is the main focus of previous work by Aronov et al. [1].

PROBLEM 8. (COLLINEARITY INDEXING WITH QUERIES ON A LINE) *Given two sets $S_1$ and $S_2$ of $n$ points on the two-dimensional integer grid $[2^w]^2$, where $w = \text{polylog}(n)$, and a vertical line $\ell$, preprocess them to answer queries of the form: given a point $q$ on $\ell$, are there two points $p_1 \in S_1$ and $p_2 \in S_2$ such that $q$, $p_1$, and $p_2$ are collinear?*

Aronov et al. showed the following result, which now also follows from a straightforward application of Lemma 3.4 and Lemma 4.3.

THEOREM 4.7. (THEOREM 1.1 OF [1]) *For any choice of a constant $0 < \alpha < 1$, we can construct in $\tilde{O}(n^2)$ time with high probability a data structure for collinearity indexing with queries on a line using $\tilde{O}(n^{2-\alpha/3})$ words of space which supports queries in $\tilde{O}(n^\alpha)$ time.*

Note that as with previous applications, our results also allow us to restrict to pairs of points lying in two boxes given at query time. Using Lemma 4.2 instead of Lemma 3.4, we could also answer queries asking for the closest line spanned by a pair of points in $S_1 \times S_2$, above or below the query point $p$, at the cost of increasing the space of our data structure to $\tilde{O}(n^{2-\alpha/4})$.

## References

[1] B. ARONOV, E. EZRA, M. SHARIR, AND G. ZIGDON, *Time and space efficient collinearity indexing*, Comput. Geom., 110 (2023), p. 101963.

[2] L. Barba, J. Cardinal, J. Iacono, S. Langerman, A. Ooms, and N. Solomon, *Subquadratic algorithms for algebraic 3SUM*, Discret. Comput. Geom., 61 (2019), pp. 698–734.
[3] S. Bespamyatnikh and M. Segal, *Selecting distances in arrangements of hyperplanes spanned by points*, J. Discrete Algorithms, 2 (2004), pp. 333–345.
[4] P. Bille, I. L. Gørtz, M. Lewenstein, S. P. Pissis, E. Rotenberg, and T. A. Steiner, *Gapped string indexing in subquadratic space and sublinear query time*. Manuscript, 2022.
[5] C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra, *A strong and easily computable separation bound for arithmetic expressions involving radicals*, Algorithmica, 27 (2000), pp. 87–99.
[6] J. Cardinal and M. Sharir, *Improved algebraic degeneracy testing*, in 39th International Symposium on Computational Geometry, SoCG 2023, June 12–15, 2023, Dallas, Texas, USA, E. W. Chambers and J. Gudmundsson, eds., vol. 258 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 22:1–22:16.
[7] B. Chazelle, *Some techniques for geometric searching with implicit set representations*, Acta Informatica, 24 (1987), pp. 565–582.
[8] H. Corrigan-Gibbs and D. Kogan, *The function-inversion problem: Barriers and opportunities*, in Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1–5, 2019, Proceedings, Part I, D. Hofheinz and A. Rosen, eds., vol. 11891 of Lecture Notes in Computer Science, Springer, 2019, pp. 393–421.
[9] M. Farach, *Optimal suffix tree construction with large alphabets*, in 38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19–22, 1997, IEEE Computer Society, 1997, pp. 137–143.
[10] A. Fiat and M. Naor, *Rigorous time/space trade-offs for inverting functions*, SIAM J. Comput., 29 (1999), pp. 790–803.
[11] A. Gajentaan and M. H. Overmars, *On a class of $O(n^2)$ problems in computational geometry*, Comput. Geom., 45 (2012), pp. 140–152.
[12] I. Goldstein, T. Kopelowitz, M. Lewenstein, and E. Porat, *Conditional lower bounds for space/time tradeoffs*, in Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 – August 2, 2017, Proceedings, F. Ellen, A. Kolokolova, and J. Sack, eds., vol. 10389 of Lecture Notes in Computer Science, Springer, 2017, pp. 421–436.
[13] A. Golovnev, S. Guo, T. Horel, S. Park, and V. Vaikuntanathan, *Data structures meet cryptography: 3SUM with preprocessing*, in Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22–26, 2020, K. Makarychev, Y. Makarychev, M. Tulsiani, G. Kamath, and J. Chuzhoy, eds., ACM, 2020, pp. 294–307.
[14] A. Grønlund and S. Pettie, *Threesomes, degenerates, and love triangles*, J. ACM, 65 (2018), pp. 22:1–22:25.
[15] J. Iacono and S. Langerman, *Dynamic point location in fat hyperrectangles with integer coordinates*, in Proceedings of the 12th Canadian Conference on Computational Geometry, Fredericton, New Brunswick, Canada, August 16–19, 2000.
[16] D. M. Kane, S. Lovett, and S. Moran, *Near-optimal linear decision trees for k-SUM and related problems*, J. ACM, 66 (2019), pp. 16:1–16:18.
[17] T. Kopelowitz and E. Porat, *The strong 3SUM-INDEXING conjecture is false*. Manuscript, 2019.
[18] U. Manber and E. W. Myers, *Suffix arrays: A new method for on-line string searches*, SIAM J. Comput., 22 (1993), pp. 935–948.
[19] D. E. Willard, *Log-logarithmic worst-case range queries are possible in space $\theta(n)$*, Inf. Process. Lett., 17 (1983), pp. 81–84.