A APPENDIX: EXPERIMENTS

Table 1 shows the number of queries for various algorithms and datasets, as used in Figure 2. Similarly, Table 2 shows the total number of annotations used, as depicted in Figure 3.

N	LOGSTRAT	PAULA	ADASTRAT
35615	78	78	18
356150	156	156	20
3561500	234	234	18
35615000	312	312	21

Table 1: Number of queries of the precision function issued by various algorithms for different size variants of PPDB-36K.

N	RANDOM	LOGSTRAT	PAULA	ADASTRAT
35615	24745	18287	11292	14612
356150	84369	35631	19092	24004
3561500	284834	53937	26892	23707
35615000	954359	72867	34692	28477

Table 2: Number of annotations used by various algorithms for different size variants of PPDB-36K.

B APPENDIX: PROOFS

B.1 Noisy Samples for Point Estimates

Consider the setting of Section 2.1, where we use noisy estimates $\tilde{v}(u)$ of $v(u)$ at J randomly chosen points from $\{1, 2, \ldots, r\}$, in order to estimate $p(r)$.

First, fix a u. Because of the error probability $\eta < 1/2$ in the estimate, the expected value $\tilde{v}(u)$ is $v(u)(1-\eta) + (1-v(u))\eta$, which equals $v(u) + \eta - 2\eta v(u)$. Second, note that the expected value of $\sum_{j\in J} v(t_j)$ across random choices of J is precisely $zp(r)$, where $z = |J|$.

Putting these together, we obtain that the expected value of $\frac{1}{z}\sum_{j\in J}\tilde{v}(t_j)$ across random choices of J is $p(r) + \eta - 2\eta p(r)$. Assuming $p(r) \le 1/3$, this lies within $(1 \pm \eta)\, p(r)$ and is also within a multiplicative factor of $1+\eta$ of $p(r)$.

Given $\epsilon > \eta$, we can now use the two-sided Hoeffding bound (Hoeffding, 1963) to assess how many samples, z, are needed to obtain a $(1+\epsilon)$-approximation of $p(r)$. For this, it suffices to use a z that guarantees a $\frac{1+\epsilon}{1+\eta}$-approximation of the expected value of $\frac{1}{z}\sum_{j\in J}\tilde{v}(t_j)$, which, as shown above, is always within a factor of $1+\eta$ of $p(r)$. To this end, a direct application of Hoeffding's inequality, while observing that $\frac{1+\epsilon}{1+\eta} - 1 = \frac{\epsilon-\eta}{1+\eta}$, yields:

$$z \ge \frac{(1+\eta)^2}{2(\epsilon-\eta)^2 p(r)^2} \ln\frac{2}{\delta}.$$

B.2 Proof of Theorem 2

The upper bound: when $v' \le y$, $p(v')v' \le p(y)y$ because of the monotonicity of the partial sum $\sum_{j=1}^{n} v(t_j)$. Therefore, $p(v') \le p(y)y/v'$, which is the result on the first line. When $v' > y + m$, $p(v') \le p(y)$ because of weak $(\tilde{r}, m)$-monotonicity, which is the result on the fourth line. In the middle, for $y + \lfloor mp(y) \rfloor < v' \le y + m$, because of weak $(\tilde{r}, m)$-monotonicity, $p(y+m) \le p(y)$, hence at most $p(y)$ fraction of the entries $v(t_{y+1}), v(t_{y+2}), \ldots, v(t_{y+m})$ can be 1. This implies that $p(v')v' \le p(y)y + \lfloor mp(y) \rfloor$, which is the third line. For $y < v' \le y + \lfloor mp(y) \rfloor$, in the worst case all $v' - y$ terms in $v(y+1), \ldots, v(v')$ are 1. Therefore, $p(v')v' \le p(y)y + v' - y$, which is the second line.

The lower bound: the first line holds because of weak monotonicity. The fourth line is again because $p(y)y \le p(v')v'$ when $v' \ge y$. In the middle, when $y - \lfloor mp(y) \rfloor \le v' < y$, at most all $y - v'$ items $v(t_{v'+1}), \ldots, v(t_y)$ are 1. Hence, $p(v')v' + y - v' \ge p(y)y$, which gives us the third line. When $y - m \le v' < y - \lfloor mp(y) \rfloor$, again, because $p(y-m) \ge p(y)$, at most $p(y)$ fraction of the elements $v(t_{y-m+1}), \ldots, v(t_y)$ can be 1. This implies that $p(v')v' + \lfloor mp(y) \rfloor \ge p(y)y$. Rearranging terms, we get the second line.

The envelope is tight because we can construct sequences of $v(t_j)$ which match the upper and lower bound exactly.

B.3 Proof of Theorem 4

The number of calls to QUERY is one plus the number of intervals $(p(l), p(r))$ which enter the stopping condition of function PR. Notice that the boundary (l, r) of every interval $(p(l), p(r))$ which enters the stopping condition must satisfy $r/l \ge 1 + \epsilon$, since otherwise its parent function call already satisfies the stopping condition $r/l \le (1+\epsilon)^2$. Aggregated across all intervals, this can happen at most $\log_{1+\epsilon}(N/\tilde{l})$ times, as claimed.

B.4 Proof of Lemma 3

Let $\tilde{l}' = \max\left\{\lceil \frac{m}{1+\epsilon} \rceil, \tilde{r}\right\}$. Using the full characterization from Theorem 2, we can prove that for $r > l > \tilde{l}'$, $p(r) \le (1+\epsilon)p(l)$.

Because of the definition of $q_1, \ldots, q_K$, we know that $q_1 \le q_2 \le \ldots \le q_K$. Moreover, we know that for $q_i < j < q_{i+1}$, $p(j) \ge p(q_i)/(1+\epsilon)$. Because of the statement in the previous paragraph, we know that $p(j) \le p(q_i)(1+\epsilon)$. Proof completes.

B.5 Proof of Lemma 4

First, the function $p(r)r$ monotonically increases, because for $l < r$, we have

$$r\,p(r) = \sum_{i=1}^{r} v(t_i) \geq \sum_{i=1}^{l} v(t_i) = l\,p(l).$$

Therefore, $p(j)j$ is sandwiched between $p(s_i)s_i$ and $p(s_{i+1})s_{i+1}$ whenever $j \in \{s_i, \ldots, s_{i+1}\}$. Because of the definition of s_{i+1}, for any $j \in \{s_i, \ldots, s_{i+1}\}$, we have $p(j)j \geq p(s_i)s_i$ and $p(j)j \leq (1+\epsilon)p(s_i)s_i$, which finishes the proof.

B.6 Proof of Theorem 6

Suppose the "optimal" algorithm calls QUERY at points $r_1, r_2, \ldots, r_{\text{OPT}}$. We begin with some relevant notation. These points split the entire range between $\tilde{l}$ and N into OPT + 1 *segments*. We call one segment of this type an *opt-segment*. We also define the following tuple: $\langle p(l), p(r), d, \text{SMALL/BIG}\rangle$, where $(p(l), p(r))$ is an interval on which the function PR called during the execution of ADASTRAT, d is the depth of this recursive call, and the fourth entry is either SMALL or BIG. It is BIG if and only if the interval $[l, r]$ covers at least one complete opt-segment. The log distance of a tuple is measured as $\log_{1+\epsilon} r/l$. Two tuples are treated as *cousins* if they are called by a single PR function as two children calls (i.e., one is $(p(l), p(c))$, while the other one is $(p(c), p(r))$). We can *merge* two cousin tuples. The result of merging is the tuple representing the parent function that called the functions represented by these two cousin tuples. Notice that the log distance of the merged tuple is 2 times that of one cousin tuple.

We start with the set Φ, made of tuples $\langle p(l), p(r), d, \text{SMALL/BIG}\rangle$ such that $(p(l), p(r))$ enters the stopping condition of function PR during the execution of the algorithm. We repeat the following *merge* operation on tuples in Φ until no tuple in Φ is tagged with SMALL: (i) Choose a tuple from Φ that has the largest depth d among those tagged with SMALL. (ii) Merge this tuple with its cousin tuple. (iii) Add the merged tuple back into Φ. We refer to the set Φ obtained after merging all SMALL tuples as Φ^E.

First, all tuples in Φ^E are tagged BIG. Therefore, each of them contains an opt-segment. Second, since the tuples in Φ^E are still from the PR algorithm, these tuples must be non-overlapping. Based on these two observations, the number of tuples in Φ^E is bounded by the number of opt-segments, which is OPT+1. How many merge operation could each tuple in Φ^E have gone through? This must be bounded by $\mathrm{O}(\log_2 \log_{1+\epsilon} N)$, since the largest possible log distance is $\log_{1+\epsilon} N$ (the entire range), and each merge operation doubles the log distance.

Now we count the number of QUERY calls that algorithm ADASTRAT makes. It is easy to see that

$$\#\text{QUERY} = |\Phi^E| + \#\text{MERGE} + 1,$$

where #MERGE is the total number of merge operations performed. This number is bounded by

$$|\Phi^E| + |\Phi^E| \cdot \max\{\#\text{MERGE for one tuple in } \Phi^E\} + 1.$$

Combined with the fact that $|\Phi^E| \leq \text{OPT} + 1$ and #MERGE for one tuple in $\Phi^E \leq \log_2 \log_{1+\epsilon} N$, we obtain the claimed bound.

B.7 Detailed Construction of F to Prove Theorem 7

Our proof works by showing that any algorithm that can identify an unknown function from F, which includes $\mathcal{A}$, must make at least J queries. To demonstrate this, our function family F is constructed such that for one separating point, there are at least two functions from F that are more than $(1+\epsilon)^2$ apart at the point. Therefore, $\mathcal{A}$ has to query all J separating points to identify one function from F.

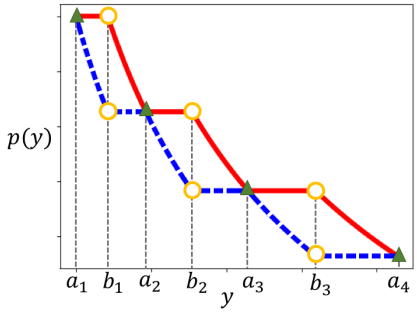

Figure 5: An illustration of the function family supporting the lower bound in Theorem 7.

Specifically, we identify $J+1$ points $a_1, \ldots, a_{J+1}$, where the space between a_j and a_{j+1} is called the j-th interval ($j = 1, \ldots, J$), which is shown as the space between two green triangles in Figure 5. One function f from F either follows the red or the blue curve in one interval. Because we have J intervals in total, this gives us 2^J functions in F. The separating points are shown in yellow circles in Figure 5. We construct these yellow circles to guarantee that the gap (height) is more than $(1+\epsilon)^2$.

We argue that Algorithm $\mathcal{A}$ must make at least one query in each of the J intervals. This is because even if $\mathcal{A}$ knows the exact function segment (red or blue) everywhere except for the j-th interval, it still cannot decide if

f follows the red or the blue curve within the j-th interval.

The exact locations of separating points are as follows. Let $a_j = \lfloor (1+\epsilon')^{4j-4} y_0 \rfloor$, $b_j = \lfloor (1+\epsilon')^{4j-2} y_0 \rfloor$, $(j = 1, \ldots, J+1)$. $N = a_{J+1} = \lfloor (1+\epsilon')^{4J} y_0 \rfloor$. Here, y_0 is a sufficiently large number. Let $1 < \gamma < {}^{(1+\epsilon')}/_{(1+\epsilon)}$. Select 2^J $(\tilde{r}, m)$-weak monotonic functions $f_0, f_1, \ldots, f_{2^J-1}$ such that their values at a_j (shown as the green triangles of Figure 5) are close. More specifically, for all functions $f_u \in \{f_0, f_1, \ldots, f_{2^J-1}\}$, their values at a_j are bounded:

$$\frac{1}{\gamma(1+\epsilon')^{2j-2}} < f_u(a_j) < \frac{\gamma}{(1+\epsilon')^{2j-2}}.$$

For a specific b_j (shown as yellow circles in Figure 5), half of the functions in $\{f_0, f_1, \ldots, f_{2^J-1}\}$ match approximately around a particular value (i.e., follow the red curve). The other half match around a different value (i.e., follow the blue curve). More specifically, for all $u \in \{0, 1, \ldots, 2^J - 1\}$, whose j-th digit of its binary representation is 0, f_u's value at b_j is bounded by

$$\frac{1}{\gamma(1+\epsilon')^{2j-2}} < f_u(b_j) < \frac{\gamma}{(1+\epsilon')^{2j-2}},$$

For all $u \in \{0, 1, \ldots, 2^J - 1\}$, whose j-th digit of its binary representation is 1, f_u's value at b_j is bounded by

$$\frac{1}{\gamma(1+\epsilon')^{2j}} < f_u(b_j) < \frac{\gamma}{(1+\epsilon')^{2j}},$$

Suppose algorithm $\mathcal{A}$ outputs an $(1+\epsilon)$-approximate curve for any $(\tilde{r}, m)$-weak monotonic precision function. Starting with one unknown function f drawn from the set $\{f_0, f_1, \ldots, f_{2^J-1}\}$, we can use algorithm $\mathcal{A}$ to identify which function f actually is. To see this, we examine the approximate values $\tilde{f}(b_1), \ldots, \tilde{f}(b_J)$ returned by algorithm $\mathcal{A}$. If

$$\tilde{f}(b_j) > \frac{1}{(1+\epsilon')^{2j-1}},$$

we know that

$$\frac{1}{\gamma(1+\epsilon')^{2j-2}} < f(b_j) < \frac{\gamma}{(1+\epsilon')^{2j-2}},$$

because otherwise, $\tilde{f}(b_j)$ and $f(b_j)$ are more than $(1+\epsilon)^2$ apart. The other side of the argument also holds.

B.8 Proof of Theorem 8

First, assuming that we have access to $r_1, \ldots, r_K$, we access the values of the first $\tilde{l}$ terms $v(t_1), \ldots, v(t_{\tilde{l}})$. We then randomly access $\lceil \frac{r_1 - \tilde{l}}{r_1} \rceil s$ items from $\tilde{v}(t_{\tilde{l}+1}), \ldots, \tilde{v}(t_{r_1})$, randomly access $\lceil \frac{r_2 - r_1}{r_2} \rceil s$ items from $\tilde{v}(t_{r_1+1}), \ldots, \tilde{v}(t_{r_2})$, and so on, until we randomly access $\lceil \frac{r_K - r_{K-1}}{r_K} \rceil s$ items from $\tilde{v}(t_{r_{K-1}+1}), \ldots, \tilde{v}(t_{r_K})$.

When we compute $\textsc{Query}(r_i, T, \tilde{v})$, we randomly subsample accessed items in the range of $1, \ldots, \tilde{l}$ with probability $\frac{r_i - r_{i-1}}{r_i}$; randomly subsample accessed items in the range of $\tilde{l}+1, \ldots, r_1$ with probability $\frac{r_i - r_{i-1}}{r_i} \frac{r_1}{r_1 - \tilde{l}}$; randomly subsample accessed items in the range of $r_1 + 1, \ldots, r_2$ with probability $\frac{r_i - r_{i-1}}{r_i} \frac{r_2}{r_2 - r_1}$; and so on; randomly subsample accessed items in the range of $r_{i-2} + 1, \ldots, r_{i-1}$ with probability $\frac{r_i - r_{i-1}}{r_i} \frac{r_{i-1}}{r_{i-1} - r_{i-2}}$; and take all accessed items in the range of $r_{i-1} + 1, \ldots, r_i$. We use the empirical mean of these subsampled items to compute the estimation $\tilde{p}(r_i)$. Using Hoeffding's inequality, Eq. (2), we know that the estimation is a β-approximation with confidence at least $1 - \delta/K$.

Next, applying the union bound over all $i \in \{1, \ldots, K\}$, we have overall confidence of at least $1 - \delta$ that the estimations are β-approximations simultaneously for all K points $r_1, \ldots, r_K$.

Nevertheless, we do not know the location of $r_1, \ldots, r_K$ upfront. We thus take an iterative deepening style adaptive sampling scheme, as detailed in the last two paragraphs of Section 4. In particular, we gradually increase the number of samples in each interval as we increase K; and we sample more points for intervals that fall below the sample density thresholds when we introduce new query points.