FROM RANKING QUALITY TO QUERY COMPLEXITY: EXACT-AUC MINIMAX RECOVERY WITH ORDINAL PREDICTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We characterize the minimax adaptive-query complexity of every exact-AUC class in noiseless group testing with a known number of positives. For $s \geq 2$ positives among $d \geq 2s$ items, let $\Delta = s(d-s)\min\{\mathrm{AUC}, 1-\mathrm{AUC}\}$. The complexity is $\Theta(\min\{\Delta, s\} + s\log(1+\Delta/s^2))$ for $\Delta \geq 2$; the classes with $\Delta \in \{0,1\}$ are singletons and require no queries when AUC is known. The curve exhibits a regime in which candidate counting is insufficient: for $2 \leq \Delta \leq s$, log-cardinality is $\Theta(\sqrt{\Delta})$, yet recovery requires $\Theta(\Delta)$ OR queries. We obtain matching lower bounds by constructing hard families at every prescribed rank sum. One explicit algorithm, given neither AUC nor a candidate class, attains all these class-wise rates up to universal constants and an additive constant. The characterization also identifies how ranking accuracy must scale with sparsity to permit $O(s)$ query budgets. Within the same exact-AUC class, however, supports recoverable by this algorithm in $O(s)$ queries can coexist with subfamilies requiring $\Omega(s\log s)$ queries in the worst case. Certified finite-class budgets and numerical constructions illustrate the query obstruction and the within-class separation.

1 INTRODUCTION

In adaptive group testing, even fixing AUC at 0.99 can leave worst-case exact recovery at the same query order as having no ranking when positives become sparse. Consider a screening pipeline that uses a learned model to rank candidates, then pooled measurements to identify all positives. The downstream question is how much measurement effort the ranking can save. AUC measures pairwise ordering (Fawcett, 2006; Menon and Williamson, 2016); exact recovery must find every positive. How accurate must a ranking be to reduce recovery cost, and can an algorithm exploit that accuracy without knowing it?

Our model has d items and an unknown set S of s positives. A noiseless OR query asks whether an arbitrary subset intersects S; the algorithm knows s and a predicted total order q, but receives no calibrated probabilities (Dorfman, 1943; Hwang, 1972; Aldridge et al., 2026). If the positive ranks are $\rho_1 < \cdots < \rho_s$, their total displacement from the first s positions is

$$D_q(S) = \sum_{i=1}^{s}(\rho_i - i) = s(d-s)(1-\mathrm{AUC}_q(S)). \tag{1}$$

Prescribing empirical AUC therefore specifies a natural candidate class: all size-s supports with one exact rank sum. We ask for the minimax query complexity of that class, allowing the competing algorithm to know its AUC value.

Because s is known, any predesignated support can be verified with one complement query. We therefore compare recovery guarantees over entire quality classes.

A sharp minimax curve for every exact-AUC class. Our main contribution is a complete characterization of the query budget at each feasible ranking-quality level. Let $M(d, s, D)$ denote the minimax budget on the exact-displacement class, with per-instance recovery probability at least $2/3$, and put $\Delta = \min\{D, s(d-s) - D\}$. For $s \geq 2$, $d \geq 2s$, and every feasible integer displacement

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

with $\Delta \geq 2$,

$$M(d, s, D) = \Theta\left(\min\{\Delta, s\} + s \log\left(1 + \frac{\Delta}{s^2}\right)\right). \tag{2}$$

The remaining classes, with $\Delta = 0$ or 1, are singletons and have zero budget. Reversing the ranking exchanges D and $s(d-s) - D$, so quality near either orientation endpoint gives a small class-wise cost. The curve grows linearly in Δ near the boundary, remains of order s between displacements s and s^2, and then grows logarithmically in Δ/s^2.

The curve makes the required accuracy depend explicitly on sparsity: an $O(s)$ class budget is attainable precisely when $\min\{\mathrm{AUC}, 1 - \mathrm{AUC}\} = O(s/(d-s))$. On the high-AUC side, ranking error must therefore decrease with the fraction of positives. Even convergence of AUC to one need not reduce the query order. For example, take $d = 2s^3$ and $D = s^3$. Then $\mathrm{AUC} = 1 - \Theta(1/s) \to 1$, yet $M(d, s, D) = \Theta(s \log s) = \Theta(s \log(d/s))$, the recovery order without a ranking. The rate at which ranking error vanishes is therefore decisive.

The technical challenge is to make the lower bounds hold at every prescribed quality level. After reversal, consider $D \leq s(d-s)/2$. For $2 \leq D \leq s$, the class contains $p(D)$ supports, where p is the partition number, so $\log |\mathcal{C}_D| = \Theta(\sqrt{D})$. Yet recovery requires $\Theta(D)$ queries. We realize this gap through disjoint, equal-sum negative pairs: every informative query has a negative branch containing at most one candidate in the constructed family. Shifted blocks and symmetric positive-rank pairs give matching bounds in the remaining regimes. Preserving the rank sum in each construction makes the lower bounds hold at every feasible AUC.

One algorithm attains the entire curve without AUC. Our explicit algorithm U combines top-s verification, geometrically expanding boundary windows, and a rank-band fallback, and alternates its queries between the two ranking directions. For every feasible D, the same algorithm satisfies

$$\sup_{S \in \mathcal{C}_D} T_U(S) = O(1 + M(d, s, D)).$$

It never enumerates the exact-AUC class. The additive constant accommodates the zero-query class-aware endpoints. The constant endpoint overhead relies on inexpensive verification: a pool-size cap can make this cost grow with s (Proposition 1).

The same AUC permits different recovery costs. We show that U also satisfies an upper bound controlled by $\Psi_q(S) = \sum_i \log(e\rho_i/i)$ and its reversed-order counterpart. For a sequence of exact-AUC classes, a support recoverable by U in $O(s)$ queries coexists with a subfamily requiring $\Omega(s \log s)$ queries in the worst case. The main curve describes the class envelope; positive-rank geometry describes variation inside it.

Related work. Within the broader study of algorithms with predictions (Lykouris and Vassilvitskii, 2018; Purohit et al., 2018), prediction-dependent query bounds arise in sorting (Bai and Coester, 2023), distribution-assisted binary search (Dinitz et al., 2024), and recent search data structures driven by predicted access frequencies (Fu et al., 2025; Chen et al., 2025). Group testing incorporates side information through prior probabilities (Li et al., 2014), candidate set-systems (Gonen et al., 2022), and correlated hypergraph and Markov models (Nikpey et al., 2024; Ravi and Shomorony, 2025). Weighted hypergraph search (Triesch, 1996) supplies our candidate-search upper bound and also implies the existence of an $O(\Psi)$ ordinal bound. Competitive group testing studies adaptation to an unknown defective count (Du and Hwang, 1993); here the count is known and ranking quality is unknown. We characterize every prescribed exact-AUC class and give an explicit algorithm attaining the curve with finer rank-dependent guarantees. Appendix D details these connections and the weighted-search and verification-and-doubling reductions.

2 Problem Setup and Quality Classes

The oracle returns $Y(Q) = \mathbf{1}\{Q \cap S \neq \varnothing\}$ for any $Q \subseteq [d]$. The unknown support has known cardinality s, with $s \geq 2$ and $d \geq 2s$. Each oracle call costs one unit, irrespective of pool size. The total order $q = (\sigma_1, \ldots, \sigma_d)$ is fixed before querying; score ties are resolved by a fixed rule. All AUC values refer to this resolved order. We identify items with rank coordinates when convenient, write $h = d - s$, and use natural logarithms unless a base is displayed.

Exact quality classes. For positive ranks $\rho_1 < \cdots < \rho_s$, define

$$D_q(S) = \sum_{i=1}^{s} (\rho_i - i), \qquad \Delta_q(S) = \min\{D_q(S), sh - D_q(S)\},$$
$$\mathcal{C}_D(q) = \{S \subseteq [d] : |S| = s,\ D_q(S) = D\}. \tag{3}$$

Exactly $\rho_i - i$ negatives precede the ith positive, giving

$$\mathrm{AUC}_q(S) = 1 - \frac{D_q(S)}{sh}. \tag{4}$$

Every integer $D \in [0, sh]$ is feasible; a constructive verification is given in Appendix A.1. We suppress q when it is fixed.

Minimax budget. Let $M(d, s, D)$ be the smallest integer t for which an adaptive algorithm, given d, s, q, D, makes at most t queries for every $S \in \mathcal{C}_D$ and every realization of its internal randomness, and satisfies $\Pr(\widehat{S} = S) \geq 2/3$ for each such S. The budget is uniform, rather than an expected query count. The asymptotic upper bounds in our main results are attained deterministically with zero error. Relabeling items makes M independent of the particular total order.

Verification, symmetry, and endpoints. For any size-s set S_0, $Y([d] \setminus S_0) = 0$ implies $S \subseteq S_0$ and hence $S = S_0$. An algorithm can perform this verification first and use a general recovery procedure if it fails. For the reversed order, $\rho'_i = d + 1 - \rho_{s+1-i}$, so

$$D_{q^{\mathrm{rev}}}(S) = sh - D_q(S), \qquad M(d, s, D) = M(d, s, sh - D). \tag{5}$$

The classes $D = 0$ and $D = 1$ consist of $[s]$ and $[s-1] \cup \{s+1\}$, respectively; reversal gives the other two singleton endpoints. Thus $M = 0$ for $\Delta \in \{0, 1\}$. The assumption $s \geq 2$ is necessary: with $s = 1$, every exact-AUC class is a singleton, while an algorithm not given AUC still faces an unrestricted single-positive search.

3 The Exact-AUC Minimax Curve

Define

$$F_s(x) = \min\{x, s\} + s \log(1 + x/s^2), \qquad x \geq 0. \tag{6}$$

Our main theorem characterizes the entire collection of quality classes.

Theorem 1 (Exact-AUC minimax complexity)**.** *For all integers $s \geq 2$, $d \geq 2s$, and $0 \leq D \leq s(d-s)$,*

$$M(d, s, D) = \begin{cases} 0, & \Delta \in \{0, 1\}, \\ \Theta(F_s(\Delta)), & \Delta \geq 2, \end{cases} \qquad \Delta = \min\{D, s(d-s) - D\}. \tag{7}$$

The implicit constants are independent of d, s, D.

Table 1 summarizes the three regimes.

Table 1: The three regimes of the exact-AUC minimax curve. Only feasible displacement ranges are included.

Displacement	Minimax scale	Lower-bound construction
$2 \leq \Delta \leq s$	$\Theta(\Delta)$	Equal-sum negative pairs
$s \leq \Delta \leq s^2$	$\Theta(s)$	Shifted dense block
$s^2 \leq \Delta \leq sh/2$	$\Theta(s \log(1 + \Delta/s^2))$	Exact-sum counting

By equation 5, it suffices to work with $0 \leq D \leq sh/2$. We give the main constructions here; the complete proof, including all endpoint and integrality checks, is in Appendix A.

3.1 UPPER BOUNDS FROM DISPLACEMENT CONSTRAINTS

Lemma 1 (Forced boundary regions)**.** *If $D_q(S) \le b \le s$ for an integer b, every rank at most $s - b$ is positive and every rank greater than $s + b$ is negative.*

Proof. A negative at rank $j \le s - b$ would precede at least $s - j + 1 \ge b + 1$ positives, already contributing more than b inversions. Also $\rho_s - s \le \sum_i (\rho_i - i) \le b$. □

For $D \le s$, only the $2D$ positions around the top-s boundary remain undetermined, so singleton testing gives $O(D)$ queries. For $D \ge s$, we use the general candidate-search guarantee $O(s+\log|\mathcal{E}|)$ for a known size-s support family $\mathcal{E}$ (Triesch, 1996); Lemma 3 gives the self-contained constant-factor form needed here. Put $R = s(s+1)/2 + D$. Every support of rank sum at most R contributes $s!$ distinct ordered tuples. There are $\binom{R}{s}$ positive-integer s-tuples with sum at most R, whence

$$|\mathcal{C}_D| \le |\mathcal{C}_{\le D}| \le \frac{\binom{R}{s}}{s!}, \qquad \log|\mathcal{C}_{\le D}| \le 2s + s\log(R/s^2). \tag{8}$$

Here $\mathcal{C}_{\le D} = \bigcup_{j=0}^{D} \mathcal{C}_j$. Since $R/s^2 \le 1 + D/s^2$, candidate search costs $O(s + s\log(1 + D/s^2)) = O(F_s(D))$ for $D \ge s$. This proves the class-aware upper bound independently of the explicit algorithm introduced in Section 4.

3.2 PRESCRIBED-RANK-SUM QUERY IMBALANCE

Consider m candidates obtained from a common active set by deleting mutually disjoint negative pairs. Every informative query has a negative branch containing at most one candidate. A deterministic depth-t tree can therefore output the correct support on at most $t + 1$ candidates. Averaging over a uniform candidate and the algorithm's random seed yields

$$t \ge \frac{2m}{3} - 1 \tag{9}$$

under the per-instance success requirement. This is the OR-query imbalance mechanism underlying dense set-system search (Gonen et al., 2022). Equal-sum negative pairs allow us to realize it at each required displacement.

Specifically, for every $2 \le D \le s$, take

$$S_j = [s+2] \setminus \{s - D + j, s + 3 - j\}, \qquad 1 \le j \le m := \lfloor (D+2)/2 \rfloor. \tag{10}$$

The deleted pairs are disjoint and each sums to $2s - D + 3$. Hence all supports have size s and displacement exactly D. Since $m = \Theta(D)$, equation 9 proves the matching $\Omega(D)$ lower bound.

For a concrete example, let $s = 6$ and $D = 4$. The three candidates are $[8]$ with, respectively, the pairs

$$\{3, 8\}, \qquad \{4, 7\}, \qquad \{5, 6\}$$

removed. All removed pairs sum to 11, so each support has rank sum $36 - 11 = 25 = 6 \cdot 7/2 + 4$. Querying $\{3\}$ isolates the first candidate only on a negative answer. A query intersecting two different removed pairs is positive on all three candidates. Equal pair sums enforce the quality constraint; disjointness enforces the unbalanced decision tree.

This obstruction exceeds the entropy scale of the full class. The nondecreasing integers $e_i = \rho_i - i$ form a partition of D after reversing and removing zeros. When $D \le s \le h$, every partition is allowed, so $|\mathcal{C}_D| = p(D)$ and $\log|\mathcal{C}_D| = \Theta(\sqrt{D})$ (NIST DLMF, 2026). The peeling subfamily prevents balanced OR-query discrimination and forces the linear lower bound.

3.3 COMPLETING THE PLATEAU AND COUNTING REGIMES

For $s \ge 3$ and every $s \le D \le sh/2$, shift an active block of $s + 2$ ranks. Set

$$a = \lfloor D/s - 1/2 \rfloor, \qquad t_0 = D - sa, \qquad K_0 = 2s + 3 - t_0.$$

Deleting any local pair of sum K_0 from $a+[s+2]$ gives a support of displacement $sa+2s+3-K_0 = D$. There are $\Omega(s)$ disjoint choices because $s/2 \le t_0 < 3s/2$. The peeling bound gives $\Omega(s)$ for every such D, matching the curve throughout $s \le D \le 4s^2$.

For $s \ge 3$ and $D \ge 4s^2$, write

$$R = s(s+1)/2 + D, \qquad K = \lfloor 2R/s \rfloor, \qquad k = \lfloor (s-1)/2 \rfloor.$$

Choose k disjoint pairs of ranks $\{j, K-j\}$. Their total rank sum is kK, regardless of which pairs are selected. If $s = 2k+1$, add the fixed rank $R - kK$. If $s = 2k+2$, use the fixed ranks

$$u = \lfloor K/2 \rfloor - 1, \qquad v = R - kK - u.$$

The fixed ranks absorb the residual sum, so every choice lies in the prescribed class. Since $9s+1 \le K \le d+1$, all symmetric pairs lie in the item range; excluding pairs touching the fixed ranks loses at most two choices. Appendix A.6 supplies valid fixed ranks and shows that at least $m \ge D/(2s)$ usable pairs remain. Therefore

$$\log |\mathcal{C}_D| \ge \log \binom{m}{k} \ge \frac{s}{4} \log(D/s^2) = \Omega\big(s \log(1 + D/s^2)\big). \tag{11}$$

A depth-t binary tree can be correct on at most 2^t candidates; averaging over random seeds gives the same lower-bound order for our randomized benchmark. For $s = 2$, the candidates are disjoint positive pairs of fixed sum and admit direct bisection (Appendix A.7). These cases complete the curve.

3.4 Quality guarantees and sparsity

The curve calibrates AUC against sparsity. Write $\varepsilon = \min\{\mathrm{AUC}, 1 - \mathrm{AUC}\}$, so $\Delta = sh\varepsilon$. The transitions occur at $\varepsilon = 1/h$ and s/h.

Corollary 1 (Quality relative to sparsity)**.** *Along feasible parameter sequences with $\Delta \ge 2$,*

$$M(d, s, D) = O(s) \quad \Longleftrightarrow \quad \varepsilon = O(s/h).$$

If $d/s \to \infty$ and $\varepsilon \ge \varepsilon_0$ for a fixed $\varepsilon_0 > 0$, then $M(d, s, D) = \Theta(s \log(d/s))$, the unrestricted size-$s$ recovery order.

For example, take $\mathrm{AUC} \ge 1/2$ and let $s \to \infty$. If $d = \Theta(s^2)$, an $O(s)$ class budget requires and is guaranteed by $1 - \mathrm{AUC} = O(1/s)$. If $d = \Theta(s^3)$, the corresponding requirement is $1 - \mathrm{AUC} = O(1/s^2)$. The same linear recovery budget therefore demands increasingly accurate rankings as the candidate pool grows relative to the positive set. The ratio $h\varepsilon/s = \Delta/s^2$ locates the transition from the linear-in-s budget to the logarithmic regime.

The result also extends to a quality threshold: let $\mathcal{C}_{\le B} = \{S : |S| = s,\ D_q(S) \le B\}$, equivalently $\mathrm{AUC}_q(S) \ge 1 - B/(sh)$, and define $M_{\le B}(d, s)$ with the same budget and success convention, giving B to the competitor.

Corollary 2 (Quality-threshold classes)**.** *For every integer $1 \le B \le sh/2$, $M_{\le B}(d, s) = \Theta(1 + F_s(B))$. The endpoints satisfy $M_{\le 0}(d, s) = 0$ and $M_{\le 1}(d, s) = 1$.*

The threshold upper bound uses the same displacement-budget arguments; $\mathcal{C}_B \subseteq \mathcal{C}_{\le B}$ supplies the lower bound for $B \ge 2$. Both corollaries are proved in Appendix A.8.

4 AUC-Oblivious Recovery

We now attain the class-wise curve using only the ranking, the known cardinality, and oracle answers. The one-direction procedure B verifies the top-s support, expands a boundary window if verification fails, and falls back to rank-band recovery. Algorithm 1 combines $B(q, s)$ and $B(q^{\mathrm{rev}}, s)$ by alternating their queries.

In the pseudocode, $q(A) = \{\sigma_i : i \in A\}$ converts rank positions to items, $[0] = \varnothing$, and $Y(\varnothing) = 0$ requires no oracle call.

The fallback rank searches geometrically growing rank bands by balanced binary splitting, stopping after a completed band has revealed all s positives. Appendix B specifies the bands and proves

$$T_{\mathrm{rank}}(q, S) = O\big(s + s \log(1 + D_q(S)/s^2)\big). \tag{12}$$

Algorithm 1 AUC-oblivious recovery $U(q, s)$

One-direction procedure $B(q, s)$
if $Y(q(\{s+1, \dots, d\})) = 0$ **then return** $q([s])$
for $j = 0, \dots, \lceil \log_2 s \rceil$ **do**
  $w \leftarrow \min\{2^j, s\}$
  if $Y(q(\{s+w+1, \dots, d\})) = 0$ **then**
    Test each singleton at ranks $s-w+1, \dots, s+w$; collect the positive ranks in P
    if $|P| = w$ **then return** $q([s-w] \cup P)$
  end if
end for
return RANK(q, s)

Two-direction schedule $U(q, s)$**.** Maintain independent states for $B(q, s)$ and $B(q^{\mathrm{rev}}, s)$. Alternate their next oracle queries, starting with q. After each answer, advance that state through local computation to its next query or return. Stop as soon as either state returns.

4.1 Boundary certification and one-direction cost

Lemma 2 (One-direction guarantee)**.** *The procedure B exactly recovers every size-s support and satisfies*

$$T_B(q, S) = O(1 + D_q(S)) \quad \textit{when } D_q(S) \le s, \qquad T_B(q, S) = O(1 + F_s(D_q(S))) \quad \textit{for all } S.$$

Proof. When the suffix is negative and the window contains w positives, the remaining $s - w$ positives must fill the prefix of size $s-w$. This certifies each window return using the known cardinality. The initial verification and the fallback are also exact.

If $1 \le D \le s$, the forced-region lemma certifies completion by the first width $w \ge D$: the prefix is entirely positive, the suffix entirely negative, and the window has exactly w positives. Summing the geometric window sizes and suffix tests gives $O(D)$ calls. The case $D = 0$ takes one call. For arbitrary D, all work before the fallback is $O(s)$. For $D > s$, combining this with equation 12 gives $O(F_s(D))$. □

4.2 Simultaneous attainment of the class-wise curve

Write $f = T_B(q, S)$ and $r = T_B(q^{\mathrm{rev}}, S)$ for the two standalone query counts. Alternation gives

$$T_U(S) = \min\{2f - 1, 2r\} \le 2\min\{f, r\}. \tag{13}$$

The schedule finds the cheaper orientation without knowing which one it is.

Theorem 2 (AUC-oblivious attainment)**.** *One deterministic algorithm U, given only q, s and the oracle, exactly recovers every size-s support and satisfies*

$$T_U(S) = O(1 + F_s(\Delta_q(S))). \tag{14}$$

Consequently, $\sup_{S \in \mathcal{C}_D} T_U(S) = O(1 + M(d, s, D))$ *simultaneously for every feasible D.*

Proof. Apply Lemma 2 in both directions and use equation 5 and equation 13. Since F_s is nondecreasing, the smaller bound is $O(1 + F_s(\Delta))$. Theorem 1 gives the final comparison, including the singleton classes. □

A perfect forward ranking takes one query; its reversal takes two under the forward-first schedule. The implementation $\mathcal{I}(A)$ answers a request without a new oracle call whenever the transcript and known cardinality imply its answer, and stops once they identify the support. Consequently, $T_{\mathcal{I}(A)}(S) \le T_A(S)$. For U, inferred requests still consume their scheduled turns. Appendix B.4 specifies the inference rules used in the numerical study.

4.3 The cost of verification

The one-query endpoint reflects the ability to verify a proposed support at once. Capping the pool size makes the cost of this operation explicit.

Proposition 1 (Verification under a pool-size cap)**.** *Let L be an integer with $1 \le L \le d$ and $S_0 = [s]$. Among deterministic algorithms given q, s, but not AUC, that recover every size-s support using queries of size at most L,*

$$\inf_A T_A(S_0) = \min\{s, \lceil (d-s)/L \rceil\}. \tag{15}$$

For randomized algorithms with per-instance success at least $2/3$, $\mathbb{E}[T_A(S_0)] = \Omega(\min\{s, \lceil (d-s)/L \rceil\})$.

The upper bound verifies either the proposed positives individually or their complement in batches. For the lower bound, too few queries leave a positive and a negative whose exchange preserves the transcript; Appendix H.1 gives the deterministic and randomized arguments. At $d = s^2$ and $L = s$, deterministic verification costs $s - 1$, while the exact-AUC-aware benchmark at AUC 1 is still zero. Thus a pool-size cap can turn constant additive adaptation cost into a quantity growing with s.

5 What AUC Determines—and What It Does Not

The exact-AUC curve fixes the class's worst-case scale. Individual positive ranks give a finer upper bound, allowing easy supports to coexist with subfamilies as hard as the full class.

5.1 A finer ordinal bound

Define the rank functional

$$\Psi_q(S) = \sum_{i=1}^{s} \log \frac{e\rho_i}{i} = s + \sum_{i=1}^{s} \log \frac{\rho_i}{i}. \tag{16}$$

It is a fine-grained ordinal upper-bound scale. If $P_i = i/\rho_i$ is precision at the ith positive retrieval, then

$$\Psi_q(S) - s = \sum_{i=1}^{s} \log(1/P_i). \tag{17}$$

The penalty $\Psi - s$ vanishes at perfect ranking. Appendix C.1 also gives an exact log-depth recall representation.

Proposition 2 (Fine-grained upper bounds)**.** *Rank-band recovery satisfies $T_{\mathrm{rank}}(q, S) = O(\Psi_q(S))$. The same algorithm U from Theorem 2 satisfies*

$$T_U(S) = O(\min\{\Psi_q(S), \Psi_{q^{\mathrm{rev}}}(S), 1 + D_q(S), 1 + sh - D_q(S)\}). \tag{18}$$

Bandwise splitting costs give the first bound. Since $\Psi \ge s$, it absorbs the boundary stage; $\Psi \le s + D$ and the small-displacement guarantee give $T_B = O(1 + D)$. Applying both bounds in each orientation proves equation 18; see Appendix C.2. The functional also has a matching interpretation on fixed geometric-profile classes (Appendix C.4).

5.2 Separation inside an exact-AUC class

Proposition 3 (Within-class separation)**.** *For every sufficiently large s, there are $d = 2s^3$, a displacement $D_0 = \Theta(s^3)$, a support $S_{\mathrm{easy}} \in \mathcal{C}_{D_0}$, and a subfamily $\mathcal{H} \subseteq \mathcal{C}_{D_0}$ such that*

1. *$T_U(S_{\mathrm{easy}}) = O(s)$ and $\Psi_q(S_{\mathrm{easy}}) = O(s)$;*
2. *the minimax query budget on $\mathcal{H}$ is $\Omega(s \log s)$, even when the algorithm is told $\mathcal{H}$;*
3. *$\Psi_q(S) = \Theta(s \log s)$ for every $S \in \mathcal{H}$.*

The common AUC tends to one and $M(d, s, D_0) = \Theta(s \log s)$.

For the hard family, place all positives in $\{s^2 + 1, \ldots, 2s^2\}$. Some exact-displacement fiber contains at least $\binom{s^2}{s}/s^3$ supports, giving the lower bound by transcript counting. For the easy support, concentrate that same displacement on the last positive: $\rho_i = i$ for $i < s$ and $\rho_s = s + D_0$. Then $\Psi = s + \log(1 + D_0/s) = O(s)$, so Proposition 2 gives the claimed recovery cost. The full construction is in Appendix C.3; Section 6.3 evaluates this construction across scales.

6 NUMERICAL ILLUSTRATIONS

We examine the curve through exact finite budgets, recovery with shared ranking information, and the easy–hard construction at identical AUC. Costs count actual oracle calls under the common inference wrapper $\mathcal{I}$ (Appendix B.4). The class optimizer receives D; the recovery algorithms receive only their order, s, and query answers.

6.1 EXACT BUDGETS AND QUERY GEOMETRY

For every $d = 4, \ldots, 9$, $2 \le s \le \lfloor d/2 \rfloor$, and feasible D, we compute the randomized budget M from Section 2 and the zero-error deterministic optimum M_{det}. This covers 156 classes and 507 support records, with at most 12 supports in a class. Every OR-query partition is enumerated; deterministic budgets and randomized feasibility are solved over candidate subsets. A separate checker verifies integer decision-tree witnesses and rational probability certificates (Appendix E).

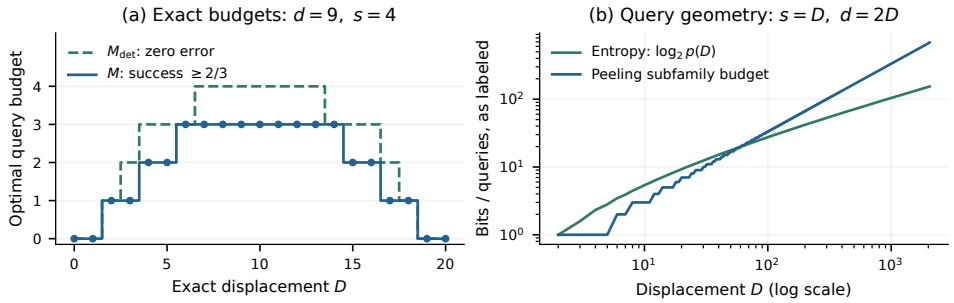


Figure 1: Exact budgets and the small-displacement obstruction. **(a)** Certified optima for every feasible D at $(d, s) = (9, 4)$: M allows per-instance error $1/3$ under a uniform budget, while M_{det} requires zero error. **(b)** Computed full-class entropy and the exact randomized budget of the peeling subfamily for $s = D$, $d = 2D$. The subfamily budget lower-bounds the full-class optimum.

Figure 1(a) exhibits reversal symmetry, zero-query singleton endpoints, and discrete differences between the two success criteria. Randomization lowers the budget in 50 of the 156 classes. For example, $(d, s, D) = (9, 4, 10)$ has 12 candidates and budgets $M = 3$ and $M_{\mathrm{det}} = 4$. These computations resolve finite constants in small classes. Appendix E.4 additionally compares the exact benchmarks with the complete-class maximum of $\mathcal{I}(U)$.

Figure 1(b) isolates why counting alone does not yield the boundary rate. For $s = D$ and $d = 2D$, the full class has $p(D)$ candidates, while the peeling subfamily has $m = \lfloor (D+2)/2 \rfloor$ candidates and exact randomized budget $\lceil 2m/3 \rceil - 1$. At $D = 2048$, these give about 153.67 bits of class entropy versus a 683-query subfamily budget. These are evaluations of the analytic construction, with its finite-budget derivation in Appendix F.6. Appendix F.3 also evaluates the proof-defined families in all three regimes, separating measured recovery costs from their analytic budget lower bounds.

6.2 RECOVERY WITH THE SAME RANKING INFORMATION

We evaluate $(d, s) = (96, 3), (160, 6), (224, 12)$ on a displacement grid spanning the endpoints, the transitions at $\Delta = s, s^2$, and the center. Classes with at most 48 supports are enumerated; larger classes use 48 uniform draws with replacement. Pairing supports with their reversals gives 1,544 records across 62 class-orientation settings. Alongside random-order GBS, we run the same GBS procedure in predicted rank order, preserving its group sizes, splitting, inference, and stopping rules. Ranked singleton scanning and rank-band recovery complete the five-algorithm comparison (Appendix F).

At all 28 evaluated settings with $\Delta \le s$, $\mathcal{I}(U)$ has lower mean cost than both GBS variants. Perfect forward and reverse orderings take one and two queries, respectively: boundary verification extracts savings beyond those obtained by ordering the GBS groups. Over the full grid, $\mathcal{I}(U)$ beats ranked GBS in 40 of 62 settings. As displacement grows, exploration and fallback costs become visible; random-order GBS has lower mean cost near the center in all three sizes.

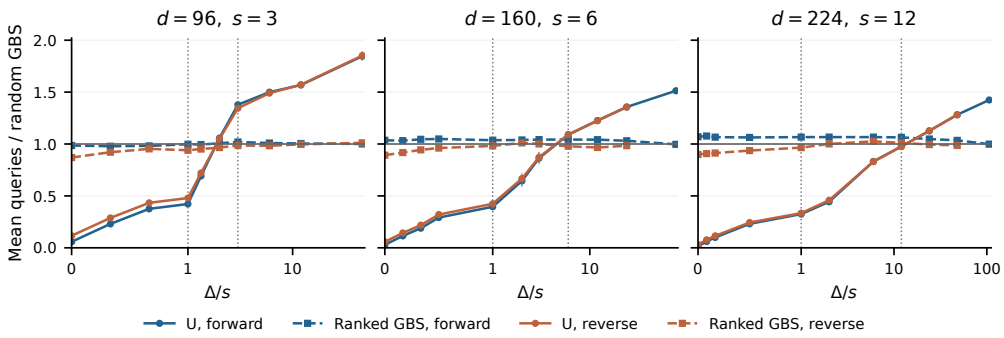


Figure 2: Recovery with shared ranking information. Solid curves show $\mathcal{I}(U)$ and dashed curves show ranked GBS, each normalized by mean random-order GBS cost (eight orders per support). Colors distinguish forward and reversed supports. Values below one improve on random-order GBS. Whiskers give pointwise 95% paired-bootstrap intervals for sampled classes; the remaining classes are enumerated. Dotted lines mark $\Delta = s, s^2$. The horizontal axis is linear up to $\Delta/s = 1$ and logarithmic thereafter.

The held-out HIV and PCBA study in Appendix G connects these costs to learned ranking quality. Across 36,000 pools, 99.925% have $\eta = \Delta/s^2 \geq 1$, and $\mathcal{I}(U)$ costs more than both GBS variants in all 12 settings. On the two PCBA assays where rank-band saves 4.51–6.71 queries over random-order GBS, forward boundary exploration adds 5.00 queries and bidirectional scheduling another 10.67–11.01 on average. On these two assays, ranked splitting saves queries, but boundary exploration and bidirectional scheduling consume that gain (Appendix G.3).

6.3 Constructed separation within exact-AUC classes

We instantiate Proposition 3 at $s \in \{4, 6, 8, 10, 12, 16\}$ with $d = 2s^3$. Within $\{s^2 + 1, \ldots, 2s^2\}$, exact integer counting identifies a largest rank-sum fiber $\mathcal{H}$ with displacement D_0. We compare 48 uniform draws from this fiber with the single support $S_{\text{easy}} = \{1, \ldots, s-1, s + D_0\}$. Both have exactly the same AUC at each size.

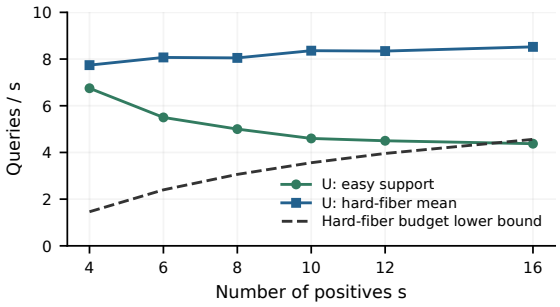


Figure 3: Same exact AUC, different recovery costs. All quantities are divided by s. Solid curves show measured $\mathcal{I}(U)$ costs: one easy support and a mean over 48 draws from its hard fiber. The dashed curve is the analytic lower bound $\log_2(2|\mathcal{H}|/3)/s$ on the fiber's uniform query budget, even when the algorithm is given $\mathcal{H}$.

The easy-support cost grows from 27 to 70 queries, while the mean on the hard fiber grows from 30.96 to 136.35. At $(d, s, D_0) = (8192, 16, 6016)$, the easy support takes 70 queries, whereas every algorithm successful with probability at least $2/3$ on each member of the hard fiber needs a uniform budget of at least 73. This gives a finite comparison between an explicit easy recovery and an algorithm-independent hard-family bound at the same quality level. Appendix F.4 gives the exact counts and sampling protocol; complete small-class comparisons appear in Appendix F.5.

Conclusion. Exact AUC determines a quality class's minimax query scale. A single AUC-oblivious algorithm attains the entire curve. The three regimes separate the roles of query geometry, positive-set size, and candidate entropy. Rank geometry explains variation within a class; the capped-pool separation identifies the role of inexpensive verification. Together, these results turn ranking quality into a quantitative recovery budget calibrated to sparsity. Further measurement settings are discussed in Appendix H.

AI USE STATEMENT

We used LLM tools during the revision and extension of this manuscript. The tools assisted with literature discovery, manuscript organization and editing, preparation of LaTeX, pseudocode, figures, and tables, and selected implementation and debugging tasks.

For the theoretical portions, AI assistance was used in discussing and refining mathematical statements, examining boundary cases, and developing and checking selected proof arguments. For the numerical studies, it assisted with experimental design, implementation of selected evaluation and synthetic-instance routines, debugging and correctness checks, and interpretation of the resulting query-cost measurements. The authors reviewed and verified the AI-assisted material, including mathematical claims and proofs, cited literature, experimental procedures, code, and reported results. The authors take responsibility for the final content of the work.

REFERENCES

Matthew Aldridge, Oliver Johnson, and Jonathan Scarlett. Group testing: An information theory perspective. Second edition. *Foundations and Trends in Communications and Information Theory*, 23(1–2):1–221, 2026. https://doi.org/10.1108/FTCIT-11-2025-0150.

Xingjian Bai and Christian Coester. Sorting with predictions. In *Advances in Neural Information Processing Systems*, volume 36, 2023. https://arxiv.org/abs/2311.00749.

Guy W. Bemis and Mark A. Murcko. The properties of known drugs. 1. Molecular frameworks. *Journal of Medicinal Chemistry*, 39(15):2887–2893, 1996. https://doi.org/10.1021/jm9602928.

Jingbang Chen, Xinyuan Cao, Alicia Stepin, and Li Chen. On the power of learning-augmented search trees. In *Proceedings of the International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 7739–7762, 2025. https://proceedings.mlr.press/v267/chen25e.html.

Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, Aidin Niaparast, and Sergei Vassilvitskii. Binary search with distributional predictions. In *Advances in Neural Information Processing Systems*, volume 37, 2024. https://arxiv.org/abs/2411.16030.

Robert Dorfman. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14(4):436–440, 1943. https://doi.org/10.1214/aoms/1177731363.

Ding-Zhu Du and Frank K. Hwang. Competitive group testing. *Discrete Applied Mathematics*, 45(3):221–232, 1993. https://doi.org/10.1016/0166-218X(93)90011-C.

Marek Eliáš, Haim Kaplan, Yishay Mansour, and Shay Moran. Learning-augmented algorithms with explicit predictors. In *Advances in Neural Information Processing Systems*, volume 37, 2024. https://arxiv.org/abs/2403.07413.

Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. https://doi.org/10.1016/j.patrec.2005.10.010.

Chunkai Fu, Brandon G. Nguyen, Jung Hoon Seo, Ryan Zesch, and Samson Zhou. Learning-augmented search data structures. In *International Conference on Learning Representations*, 2025. https://openreview.net/forum?id=N4rYbQowE3.

Mira Gonen, Michael Langberg, and Alex Sprintson. Group testing on general set-systems. In *IEEE International Symposium on Information Theory*, pp. 874–879, 2022. https://arxiv.org/abs/2202.04988.

Frank K. Hwang. A method for detecting all defective members in a population by group testing. *Journal of the American Statistical Association*, 67(339):605–608, 1972. https://doi.org/10.1080/01621459.1972.10481257.

Tongxin Li, Chun Lam Chan, Wenhao Huang, Tarik Kaced, and Sidharth Jaggi. Group testing with prior statistics. In *IEEE International Symposium on Information Theory*, pp. 2346–2350, 2014. https://arxiv.org/abs/1401.3667.

Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *Proceedings of the International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3296–3305, 2018. https://proceedings.mlr.press/v80/lykouris18a.html.

Aditya Krishna Menon and Robert C. Williamson. Bipartite ranking: A risk-theoretic perspective. *Journal of Machine Learning Research*, 17(195):1–102, 2016. https://jmlr.org/papers/v17/14-265.html.

Hesam Nikpey, Saswati Sarkar, and Shirin Saeedi Bidokhti. Group testing with general correlation using hypergraphs. *arXiv preprint arXiv:2412.17751v1*, 2024. https://arxiv.org/abs/2412.17751v1.

NIST Digital Library of Mathematical Functions. Section 27.14: Unrestricted partitions, equation 27.14.8. Accessed September 13, 2026. https://dlmf.nist.gov/27.14.E8.

Robert D. Nowak. The geometry of generalized binary search. *IEEE Transactions on Information Theory*, 57(12):7893–7906, 2011. https://arxiv.org/abs/0910.4397.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems*, volume 31, 2018. https://proceedings.neurips.cc/paper/2018/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html.

Aditya Narayan Ravi and Ilan Shomorony. Fundamental limits of non-adaptive group testing with Markovian correlation. *arXiv preprint arXiv:2501.12588*, 2025. https://arxiv.org/abs/2501.12588.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010. https://doi.org/10.1021/ci100050t.

Eberhard Triesch. A group testing problem for hypergraphs of bounded rank. *Discrete Applied Mathematics*, 66(2):185–188, 1996. https://doi.org/10.1016/0166-218X(95)00120-G.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9:513–530, 2018. https://doi.org/10.1039/C7SC02664A.

A Proof of the Exact-AUC Minimax Theorem

We prove Theorem 1 under the uniform randomized-budget convention of Section 2. Throughout, $h = d - s \geq s \geq 2$. The upper bounds are deterministic; the lower bounds allow per-instance error probability $1/3$.

A.1 Feasible displacements, reversal, and endpoints

The coordinates $e_i = \rho_i - i$ satisfy $0 \leq e_1 \leq \cdots \leq e_s \leq h$ and $\sum_i e_i = D$. Conversely, any such sequence gives a valid support through $\rho_i = i + e_i$. To realize an arbitrary integer $D \in [0, sh]$, write $D = as + b$ with $0 \leq b < s$. Take $s - b$ entries equal to a and b equal to $a + 1$. If $b > 0$, then $a < h$; if $D = sh$, all entries equal h. Thus the construction is valid throughout the range.

The reversed ranks are $\rho'_i = d + 1 - \rho_{s+1-i}$. Direct summation gives

$$\sum_{i=1}^{s} (\rho'_i - i) = s(d+1) - \sum_{i=1}^{s} \rho_i - \frac{s(s+1)}{2} = sh - D.$$

Reversal bijects the two candidate classes and relabels every allowed query, preserving its answer. It therefore preserves both budgets and per-instance success probabilities, proving equation 5.

If $D = 0$, all displacements vanish. If $D = 1$, their monotonicity forces $e_1 = \cdots = e_{s-1} = 0$ and $e_s = 1$. These give the two singleton supports stated in Section 2; reversal handles the other endpoints. The lower-bound constructions below also show that every class with $\Delta \geq 2$ has at least two members. A zero-query randomized algorithm cannot succeed with probability $2/3$ on two distinct supports, since its output probabilities must sum to at most one.

A.2 Candidate search and rank-sum counting

The following elementary version of the weighted-search principle suffices for our upper bounds. Its constant factor is immaterial; the sharper weighted hypergraph-search result is due to Triesch (1996).

Lemma 3 (Weighted candidate search)**.** *Let $\mathcal{E}$ be a known finite family of size-s supports and let $p(S) > 0$ be arbitrary design weights summing to one over $\mathcal{E}$. There is a deterministic adaptive OR algorithm, correct on every candidate, such that*

$$T(S) \leq s + \frac{\log(1/p(S))}{\log(4/3)}.$$

Proof. Maintain the design distribution conditional on all observed answers. Exclude previously identified positive items from future pools, and stop once s positive items have been identified. If an unknown item has conditional positive probability at least $1/4$, test it individually. A positive answer discovers a new positive; a negative answer removes at least a quarter of the conditional candidate mass.

Otherwise every unknown item has conditional positive probability less than $1/4$. Add unknown items to a pool until its conditional probability of a positive answer first reaches $1/4$. This is possible because every remaining candidate has an as yet unidentified positive. Each addition increases the pool's positive probability by less than $1/4$, so at stopping this probability is in $[1/4, 1/2)$. Either answer leaves at most $3/4$ of the conditional candidate mass.

At most s tests can discover new positives in the first case. Every other test multiplies the true candidate's conditional weight by at least $4/3$. That weight starts at $p(S)$ and never exceeds one, so there are at most $\log(1/p(S))/\log(4/3)$ such tests. No distributional assumption about the actual support was used. □

Uniform weights give $T = O(s + \log|\mathcal{E}|)$. The algorithm above may maintain an explicitly large candidate family; this lemma addresses query complexity.

A.3 FORCED-REGION UPPER BOUND AND COMPLETION OF THE UPPER HALF

For a known integer displacement $D \le s$, query the singleton ranks in $\{s - D + 1, \dots, s + D\}$. Lemma 1 fixes all other labels, so this is an exact algorithm using at most $2D$ queries. The singleton classes can instead be returned without any query.

For $D \ge s$, set $R = s(s+1)/2 + D$ and use the uniform version of Lemma 3 on $\mathcal{C}_D$. To justify the counting bound, the number of ordered tuples $(x_1, \dots, x_s)$ of positive integers with $\sum_i x_i \le R$ is $\binom{R}{s}$: subtract one from each coordinate, add a nonnegative slack coordinate, and apply the stars-and-bars identity. Each unordered size-s support of sum at most R accounts for $s!$ distinct such tuples. Hence

$$|\mathcal{C}_{\le D}| \le \frac{\binom{R}{s}}{s!} \le \left(\frac{e^2 R}{s^2}\right)^s.$$

Since $R/s^2 \le 1 + D/s^2$,

$$T(S) = O(s + \log|\mathcal{C}_D|) = O\big(s + s\log(1 + D/s^2)\big) = O(F_s(D)).$$

For $2 \le D \le s$, $D \le F_s(D) \le D + D/s \le 3D/2$. Thus the forced-region and candidate-search procedures give the desired upper bound on $0 \le D \le sh/2$; reversal gives the full range. The same arguments apply to $\mathcal{C}_{\le b}$ with a known budget b: use forced regions for $b \le s$ and candidate counting for $b > s$. This nested-class fact will also be used in Appendix D.4.

A.4 PRESCRIBED-SUM PEELING AND THE SMALL-DISPLACEMENT REGIME

Lemma 4 (Disjoint negative pairs)**.** *Let $\mathcal{E} = \{S_1, \dots, S_m\}$ be a known candidate family. Suppose there are disjoint sets A, P and mutually disjoint pairs $N_1, \dots, N_m \subseteq A$ with $S_j = P \cup (A \setminus N_j)$. All remaining items are negative. Any adaptive randomized algorithm with uniform query budget t and success probability at least $2/3$ on each candidate satisfies $t \ge 2m/3 - 1$.*

Proof. If a query intersects P, it is positive for every candidate. If it avoids P and A, it is negative for every candidate. Otherwise a negative answer requires the nonempty intersection with A to lie in N_j. Disjointness allows at most one such j.

Fix the internal randomness of the algorithm. Every informative node of its decision tree has at most one candidate on the negative branch. The positive branch continues the same structure. A tree of depth at most t can have at most t candidate-containing negative leaves and one final candidate-containing leaf on the continuing branch. Each leaf has a fixed output, hence the tree is correct on at most $t+1$ candidates. This remains true if the algorithm stops early or makes errors. Averaging over a uniform candidate and then over the internal randomness gives average success at most $(t+1)/m$. The assumed per-candidate guarantee makes this average at least $2/3$. For $m \ge 2$, the integer budget also satisfies $t \ge m/3$: this follows directly for $m \ge 3$, and $m = 2$ requires at least one query. □

This lemma is a variant of the dense-family query imbalance used in Gonen et al. (2022). Negative pairs, rather than single omitted positions, allow the constructions below to hold the rank sum fixed.

Small displacement and full-class entropy. For $2 \le D \le s$, use equation 10 with $m = \lfloor (D + 2)/2 \rfloor$. All ranks lie in $[s+2] \subseteq [d]$. For $1 \le j \le m$, the two deleted positions are distinct because $2j \le D + 2 < D + 3$. As j increases, the first deleted position increases and the second decreases, without their ranges overlapping. Thus the pairs are mutually disjoint. Their sums equal $2s - D + 3$ and

$$D(S_j) = \frac{(s+2)(s+3)}{2} - (2s - D + 3) - \frac{s(s+1)}{2} = D.$$

Lemma 4 gives $\Omega(D)$ for this subfamily and hence for $\mathcal{C}_D$. Together with Lemma 1, this proves the small-displacement regime, including the constant-size endpoint $D = 2$.

For the entropy statement, $e_i = \rho_i - i$ satisfies $0 \le e_1 \le \cdots \le e_s \le h$ and $\sum_i e_i = D$. Reversing the positive parts gives a partition of D into at most s parts, each at most h. Conversely, padding

any such partition with zeros and reversing it reconstructs a unique support. If $D \le s \le h$, neither restriction excludes a partition. Therefore $|\mathcal{C}_D| = p(D)$. The classical asymptotic

$$p(D) \sim \frac{1}{4D\sqrt{3}} \exp\left(\pi\sqrt{\frac{2D}{3}}\right)$$

implies $\log|\mathcal{C}_D| = \Theta(\sqrt{D})$ as D grows in this range (NIST DLMF, 2026). The singleton values $D = 0, 1$ are handled separately.

A.5 A LINEAR LOWER BOUND AT EVERY LARGER DISPLACEMENT

Assume $s \ge 3$ and $s \le D \le sh/2$. Set

$$a = \left\lfloor \frac{D}{s} - \frac{1}{2} \right\rfloor, \quad t = D - sa, \quad N = s + 2, \quad K = 2s + 3 - t.$$

Then $a \ge 0$, $s/2 \le t < 3s/2$, and $a \le \lfloor (h-1)/2 \rfloor$. Since $h \ge s \ge 3$, $a + N \le s + h = d$, so the active interval $a + [N]$ is valid.

For each local pair $1 \le u < v \le N$ with $u + v = K$, define $S_{u,v} = (a + [N]) \setminus \{a + u, a + v\}$. It has exactly s elements, and

$$D(S_{u,v}) = sa + \frac{N(N+1)}{2} - K - \frac{s(s+1)}{2} = sa + 2s + 3 - K = sa + t = D.$$

Pairs with a common sum are disjoint. Their number is

$$m = \left\lfloor \frac{N - |K - (N+1)|}{2} \right\rfloor.$$

Here $|K - (N+1)| = |s - t| \le \lfloor s/2 \rfloor$, yielding $m = \Omega(s)$ and $m \ge 2$. Lemma 4 gives $\Omega(s)$ for every integer D in the stated range. This matches $F_s(D)$ whenever $s \le D \le 4s^2$, since $\log(1 + D/s^2) \le \log 5$ there.

A.6 LARGE DISPLACEMENT: AN EXACT FAMILY FOR EACH RANK SUM

Assume $s \ge 3$ and $4s^2 \le D \le sh/2$. Put

$$R = \frac{s(s+1)}{2} + D, \qquad K = \left\lfloor \frac{2R}{s} \right\rfloor, \qquad k = \left\lfloor \frac{s-1}{2} \right\rfloor.$$

The parameter bounds give $9s + 1 \le K \le d + 1$. Start with the disjoint symmetric pairs

$$\{j, K - j\}, \qquad 1 \le j \le \left\lfloor \frac{K-1}{2} \right\rfloor.$$

Each pair lies in $[K-1] \subseteq [d]$ and has rank sum K.

Odd s. Write $s = 2k+1$. Use the fixed positive rank $z = R - kK$. The inequalities $K \le 2R/s < K + 1$ imply $K/2 \le z < (K+s)/2$. Since $K \ge 9s + 1$, this is a valid integer in $[K-1]$. Remove the pair containing z, if there is one, and choose any k remaining pairs. The resulting support has size $2k + 1 = s$ and sum R.

Even s. Write $s = 2k + 2$, put $W = R - kK$, and use fixed ranks

$$u = \lfloor K/2 \rfloor - 1, \qquad v = W - u.$$

Because s is even, $R = sK/2 + r$ with integer $0 \le r < s/2$. Thus $v = \lceil K/2 \rceil + r + 1$, so u, v are distinct and belong to $[K-1]$. Remove any pair containing either rank, at most two pairs, and choose any k remaining pairs. The support size and rank sum are again exactly s and R.

Counting and randomization. In either case, the number m of available pairs satisfies

$$m \geq \left\lfloor \frac{K-1}{2} \right\rfloor - 2 \geq K/2 - 3 \geq D/(2s).$$

For the last inequality, $K \geq s + 2D/s$ and $D \geq 4s^2$ suffice. Also $s/4 \leq k \leq s/2$. Different choices of pairs give different supports, so

$$\log |\mathcal{C}_D| \geq \log \binom{m}{k} \geq k \log(m/k) \geq \frac{s}{4} \log(D/s^2).$$

Since $D/s^2 \geq 4$, this is $\Omega(s \log(1 + D/s^2))$. No selection of a favorable displacement was used: the fixed ranks correct the sum for the given integer D.

For a uniform distribution on these $N_0 = \binom{m}{k}$ candidates, a deterministic depth-t binary decision tree has at most 2^t leaves and is correct on at most that many candidates. Fixing and then averaging the randomness proves $2^t/N_0 \geq 2/3$, hence $t \geq \log_2 N_0 + \log_2(2/3)$. This gives the required uniform bounded-error lower bound.

A.7 Two positives and completion of the proof

After reversal, $0 \leq D \leq d - 2$. Candidate supports are exactly the pairs $\{u, v\}$ with $u < v$ and $u + v = D + 3$. They are mutually disjoint, and their number is $m = \lfloor (D + 2)/2 \rfloor$. To query an arbitrary subcollection of candidates, pool one representative from each pair in that subcollection. The answer identifies whether the true pair belongs to it. Binary search is therefore possible, giving exact deterministic optimum $\lceil \log_2 m \rceil$. For $m \geq 2$, transcript counting and this upper bound imply randomized budget $\Theta(\log m) = \Theta(F_2(D))$. If $m = 1$, then $D = 0$ or 1.

The small-displacement construction, the shifted dense construction, the large exact-sum construction, and this two-positive case cover all $0 \leq D \leq sh/2$. Reverse to cover all remaining D. The upper bounds in Appendix A.3 and the singleton argument then complete the proof of Theorem 1.

A.8 Quality calibration and threshold classes

Proof of Corollary 1. For every nondegenerate class, Theorem 1 and $\Delta = sh\varepsilon$ give

$$\frac{M(d, s, D)}{s} = \Theta\left(\min\{h\varepsilon, 1\} + \log\left(1 + \frac{h\varepsilon}{s}\right)\right).$$

The first term is at most one. The logarithmic term is uniformly bounded if and only if $h\varepsilon/s$ is uniformly bounded, proving the $O(s)$ equivalence along feasible sequences.

If $d/s \to \infty$, then $h/s = d/s - 1 \to \infty$. Under $\varepsilon \geq \varepsilon_0 > 0$, with $\varepsilon \leq 1/2$,

$$\log(1 + \varepsilon_0 h/s) \leq \log(1 + h\varepsilon/s) \leq \log(1 + h/(2s)).$$

Both endpoints have order $\log(d/s)$, proving the second claim. Constants in this assertion may depend on the fixed ε_0. For comparison, unrestricted size-s recovery has order $s \log(d/s)$: uniform candidate search gives $O(s + \log \binom{d}{s})$, and bounded-error transcript counting gives $\Omega(\log \binom{d}{s})$, with $\log \binom{d}{s} = \Theta(s \log(d/s))$ for $d \geq 2s$. □

Proof of Corollary 2. The class $\mathcal{C}_{\leq 0}$ contains only $[s]$. The class $\mathcal{C}_{\leq 1}$ consists of $[s]$ and $[s-1] \cup \{s+1\}$. Querying the singleton $\{s\}$ distinguishes them. A zero-query randomized procedure cannot put output probability at least $2/3$ on both supports, so its minimax budget is exactly one.

For $2 \leq B \leq s$, the forced-region argument in Appendix A.3 gives a deterministic $2B$ upper bound uniformly over $\mathcal{C}_{\leq B}$. For $B \geq s$, the rank-sum count and uniform candidate search in that appendix give $O(s + s \log(1 + B/s^2))$. Together these are $O(F_s(B))$. The subclass $\mathcal{C}_B$ supplies a matching lower bound by Theorem 1, since $B \leq sh/2$. The constants are uniform in d, s, B. The $B = 1$ value completes the stated form with $1 + F_s(B)$. □

The same U also competes simultaneously on these threshold classes. For $S \in \mathcal{C}_{\le B}$, $\Delta_q(S) = D_q(S) \le B$, so Theorem 2 gives

$$\sup_{S \in \mathcal{C}_{\le B}} T_U(S) = O(1 + F_s(B)) = O(1 + M_{\le B}(d, s)).$$

No threshold is supplied to U.

B Explicit Algorithms and Query Accounting

This appendix specifies the fallback and the query schedule used in Algorithm 1. All procedures operate on the ranking and oracle answers. The cardinality s is known; occupancies and quality values are not inputs.

B.1 Rank-band recovery

Algorithm 2 completes each occupied band's recursive search before checking whether s positives have been recovered. Its raw splitting subroutine queries both children of a positive nonsingleton pool; answer inference is treated separately in Appendix B.4.

Algorithm 2 Rank-band recovery $\mathrm{rank}(q, s)$

Require: Ranking q, known cardinality s, and OR oracle Y

function RANK(q, s)
  $K \leftarrow \min\{r \ge 0 : 2^{r+1}s \ge d\}$
  $L_r \leftarrow \min\{2^{r+1}s, d\}$ for $r = 0, \dots, K$
  $B_0 \leftarrow [L_0]$
  for $r = 1, \dots, K$ **do**
    $B_r \leftarrow \{L_{r-1} + 1, \dots, L_r\}$
  end for
  if $K \ge 1$ and $|B_K| < |B_{K-1}|$ **then**
    Merge B_K into B_{K-1} and discard B_K
  end if
  Write the resulting bands as $A_0, \dots, A_J$
  $P \leftarrow \varnothing$
  for $r = 0, \dots, J$ **do**
    $P \leftarrow P \cup$ SPLIT($q(A_r)$)
    if $|P| = s$ **then**
      return P
    end if
  end for
end function

function SPLIT(V)
  if $V = \varnothing$ **then**
    return $\varnothing$
  end if
  if $Y(V) = 0$ **then**
    return $\varnothing$
  end if
  if $|V| = 1$ **then**
    return V
  end if
  Split V in ranking order into halves V_0, V_1 differing in size by at most one
  $P_0 \leftarrow$ SPLIT(V_0)
  $P_1 \leftarrow$ SPLIT(V_1)
  return $P_0 \cup P_1$
end function

B.2 Band costs and displacement bounds

Let $m_r = |A_r|$ and $s_r = |A_r \cap S|$ for the balanced bands in Algorithm 2. Define

$$\Phi_q(S) = 1 + \sum_{r:s_r>0} s_r \log(em_r/s_r). \tag{19}$$

This quantity captures the occupancy-dependent terms in the bandwise upper-bound accounting. Neither Φ nor Ψ records the exact within-band splitting paths.

Lemma 5 (Rank-band bounds)**.** *Let $s_r = |S \cap A_r|$ and $m_r = |A_r|$ for the balanced bands. Then*

$$T_{\mathrm{rank}}(q, S) = O(\Phi_q(S)), \qquad \Phi_q(S) = \Theta(\Psi_q(S)),$$

and

$$T_{\mathrm{rank}}(q, S) = O\big(s + s\log(1 + D_q(S)/s^2)\big).$$

Proof. Balanced recursive splitting on m items with k positives uses $O(1 + k\log(em/k))$ tests. Indeed, at depth ℓ at most $\min\{2^\ell, k\}$ nodes are positive and generate children. Sum over depths up to $\lceil \log_2 m \rceil$, separating depths before and after $\log_2 k$. For $k = 0$, one query suffices.

Let $r^\star$ be the last occupied band. Summing these local costs gives

$$T_{\mathrm{rank}} = O\left(1 + r^\star + \sum_{r \le r^\star : s_r > 0} s_r \log(em_r/s_r)\right).$$

Terminal balancing ensures $m_r = \Theta(2^r s)$, with uniform constants including the first band. Since $1 \le s_{r^\star} \le s$, the last occupied band contributes $\Omega(1 + r^\star)$ and absorbs all empty scans. Thus $T_{\mathrm{rank}} = O(\Phi)$.

If positive rank ρ_i is in A_r, then $\rho_i = O(m_r)$. Let $i_{r,1} < \cdots < i_{r,s_r}$ be the global positive indices in that band. As $i_{r,j} \ge j$,

$$\sum_{j=1}^{s_r} \log \frac{e\rho_{i_{r,j}}}{i_{r,j}} \le \sum_{j=1}^{s_r} \log \frac{Cem_r}{j} = O\left(s_r \log \frac{em_r}{s_r}\right).$$

Summing proves $\Psi = O(\Phi)$.

For the converse, $m_r = O(2^r s)$ yields

$$\Phi = O\left(s + \sum_r r s_r + \sum_{r:s_r>0} s_r \log(s/s_r)\right).$$

Put $p_r = s_r/s$ and compare it with the geometric distribution $g_r = 2^{-(r+1)}$ on the nonnegative integers. Nonnegativity of relative entropy gives $\sum_r p_r \log(1/p_r) \le \sum_r p_r \log(1/g_r) = O(1 + \sum_r r p_r)$. For $r \ge 1$, every rank in A_r exceeds $2^r s$ and $i \le s$, so its contribution to Ψ is $\Omega(1 + r)$; every contribution is at least one. Hence $s + \sum_r r s_r = O(\Psi)$ and $\Phi = O(\Psi)$.

Finally, write $R = s(s+1)/2 + D$. AM–GM and $s! \ge (s/e)^s$ imply

$$\Psi = s + \sum_i \log \rho_i - \log(s!) \le 2s + s\log(R/s^2) \le 2s + s\log(1 + D/s^2).$$

Combine this with $T_{\mathrm{rank}} = O(\Psi)$. □

B.3 Boundary certification, cost, and scheduling

Write $W_w = [s + w] \setminus [s - w]$ and $H_w = [d] \setminus [s + w]$. If $Y(H_w) = 0$ and singleton testing finds exactly w positives in W_w, precisely $s - w$ positives remain in $[s - w]$. Every rank in that prefix is therefore positive. This certificate uses the known cardinality and the observed answers, irrespective of the actual displacement. Initial top-s verification is the same argument with an empty window. If neither certificate succeeds, rank-band recovery returns the full support. This proves correctness of B.

For $1 \le D \le s$, let w_* be the first tested width at least D. Lemma 1 fixes the prefix $[s - w_*]$ and suffix H_{w_*}, so the window contains exactly w_* positives and the algorithm terminates by that stage. The width sequence is $1, 2, 4, \ldots$, truncated at s. Its sum through w_* is less than $3w_*$, even when the last step is truncated, and $w_* < 2D$. Each stage has at most one suffix query and $2w$ singleton queries. The total cost is therefore $O(D)$. For $D = 0$, the first query succeeds. With no displacement restriction, the sum of all widths is less than $3s$, so the preliminary work is $O(s)$. Lemma 5 then proves the one-direction guarantee in Lemma 2.

For the raw procedure U, the two states have standalone counts $f, r \ge 1$. The jth forward query is the $(2j-1)$st global query; the jth reverse query is the $2j$th. After each answer, the relevant state is advanced through all nonquery operations, including an empty suffix and any return. Thus the first completion occurs at $\min\{2f - 1, 2r\}$, proving equation 13. Independent local states are sufficient for this identity; the inference implementation below can additionally share facts without changing the logical schedule.

B.4 Sound inference preserves the guarantees

An implementation can infer some requested answers from previous ones. For an adaptive exact-recovery algorithm A, define $\mathcal{I}(A)$ by simulating its logical queries, omitting an oracle call only when its answer is implied by the observed transcript and the known cardinality. It may terminate early only when these facts uniquely identify a size-s support.

Lemma 6 (Query nonincrease under sound inference)**.** *For every support S, and for each fixed random seed when A is randomized, $\mathcal{I}(A)$ is correct whenever A is correct and*

$$T_{\mathcal{I}(A)}(S) \le T_A(S).$$

In particular, $\mathcal{I}(U)$ is AUC-oblivious and inherits Theorem 2 and Proposition 2.

Proof. Every inferred answer equals the true oracle answer. Induction on logical steps shows that the simulated transcript is a prefix of the transcript of A on the same support and seed. Each step makes at most one actual oracle call. If early termination occurs, the unique support consistent with the facts is necessarily the true support. Otherwise the simulation reaches the same output as A. This proves both assertions. □

The following partial inference rules suffice for a direct implementation. Maintain known-positive items P, known-negative items N, and a family $\mathcal{H}$ of positive-pool clauses. Initially $P = N = \varnothing$ and $\mathcal{H} = \{[d]\}$ because $s \ge 1$.

1. A negative answer on Q adds every item in Q to N; a positive answer adds the clause $Q \in \mathcal{H}$.
2. Remove known negatives from each positive clause. A singleton remainder identifies a positive item. A clause intersecting P is satisfied and can be discarded.
3. If $|P| = s$, all remaining items are negative. If the unclassified set has size $s - |P|$, all its items are positive.
4. A requested pool is negative if it is contained in N, and positive if it intersects P or contains an unsatisfied positive clause.

Iterate the fact updates to closure. Each rule follows directly from the OR semantics and $|S| = s$, so the procedure is sound; completeness of inference is not needed. For U, both directions share $P, N, \mathcal{H}$ but retain separate algorithmic states. An inferred answer still consumes its turn in the logical alternation. Scheduling by actual oracle calls instead would define a different algorithm and would not be covered by the prefix-of-transcript proof above.

C Fine-Grained Geometry and Separation

We develop the finer structure of Section 5: retrieval identities, algorithmic bounds, separation within an exact-AUC class, and two additional class interpretations.

C.1 PRECISION AND LOG-DEPTH RECALL IDENTITIES

Let $P_i = i/\rho_i$, $N_q(t) = |\{i : \rho_i \le t\}|$, and $N_{\mathrm{ideal}}(t) = \min\{s, \lfloor t \rfloor\}$ for $1 \le t \le d$. The precision identity equation 17 follows from $\log(\rho_i/i) = \log(1/P_i)$. Integrating that same term gives

$$\begin{aligned}\Psi_q(S) - s &= \sum_{i=1}^{s} \int_i^{\rho_i} \frac{dt}{t} \\ &= \int_1^d \frac{N_{\mathrm{ideal}}(t) - N_q(t)}{t}\, dt.\end{aligned} \tag{20}$$

Indeed, the integrand numerator counts indices with $i \le t < \rho_i$, apart from irrelevant endpoint conventions. Dividing by s expresses $(\Psi - s)/s$ as the area between ideal and realized normalized recall curves on a logarithmic depth axis. Equivalently, $\exp(-(\Psi - s)/s)$ is the geometric mean of the retrieval precisions.

C.2 THE FINE-GRAINED ALGORITHMIC BOUND

Lemma 5 proves $T_{\mathrm{rank}} = O(\Phi) = O(\Psi)$ and $\Phi = \Theta(\Psi)$. Every preliminary query of B is accounted for by the $O(s)$ bound in Appendix B.3. Since $\Psi \ge s$, this gives $T_B(q, S) = O(\Psi_q(S))$ whether the procedure returns early or uses the fallback. For $D \le s$, Lemma 2 gives $T_B = O(1+D)$. For $D > s$, set $e_i = \rho_i - i \ge 0$ and use

$$\Psi = s + \sum_i \log(1 + e_i/i) \le s + \sum_i e_i = s + D \le 2D.$$

Hence, for every support,

$$T_B(q, S) = O\big(\min\{\Psi_q(S), 1 + D_q(S)\}\big).$$

Apply this in both orientations and use equation 13 to obtain Proposition 2. These are bounds on realized algorithm cost; the identities do not impose an ordering on the exact costs of individual splitting paths.

C.3 PROOF OF THE WITHIN-CLASS SEPARATION

Proof of Proposition 3. Let $d = 2s^3$ and let $I = \{s^2+1, \dots, 2s^2\}$. For size-$s$ subsets of I, the minimum displacement is s^3 and the maximum is $2s^3 - s^2$. There are at most s^3 possible displacement values, so some D_0 has a fiber

$$\mathcal{H} = \{S \subseteq I : |S| = s,\ D(S) = D_0\}, \qquad |\mathcal{H}| \ge \binom{s^2}{s} / s^3.$$

For sufficiently large s, $\log|\mathcal{H}| = \Omega(s \log s)$: $\binom{s^2}{s} \ge s^s$ suffices. Uniform bounded-error transcript counting gives the claimed hard-family lower bound even when $\mathcal{H}$ is supplied.

For every member, $\rho_i \ge s^2 + i$, so $\rho_i/i \ge s + 1$. Conversely $\rho_i \le 2s^2$ and the factorial bound give $\Psi = O(s \log s)$. Thus $\Psi = \Theta(s \log s)$ throughout $\mathcal{H}$.

Now set $\rho_i = i$ for $i < s$ and $\rho_s = s + D_0$. The last rank is valid because $s + D_0 \le 2s^3 - s^2 + s \le d$. This support has exactly the same displacement, but $\Psi = s + \log(1 + D_0/s) = O(s)$. Proposition 2 therefore gives $T_U = O(s)$ on this support.

Finally $D_0 = \Theta(s^3)$ and $sh = 2s^4 - s^2$, so the common $1 - \mathrm{AUC} = D_0/(sh) = \Theta(1/s)$ tends to zero. For sufficiently large s, $D_0 \le sh/2$, and Theorem 1 gives $M(d, s, D_0) = \Theta(s \log s)$. □

C.4 FIXED GEOMETRIC-PROFILE CLASSES

For the balanced bands, fix a feasible occupancy vector $\mathbf{s} = (s_0, \dots, s_J)$ with $\sum_r s_r = s$, and let

$$\mathcal{C}(\mathbf{s}) = \{S : |S \cap A_r| = s_r \text{ for every } r\}, \qquad \Phi(\mathbf{s}) = 1 + \sum_{r : s_r > 0} s_r \log(e m_r / s_r).$$

Proposition 4 (Fixed geometric profiles)**.** *Under $d \geq 2s$, the minimax query budget on $\mathcal{C}(\mathbf{s})$ is $\Theta(\Phi(\mathbf{s}))$, even when the competitor is told the entire profile. Rank-band recovery, which is not told the profile, attains the upper bound. The same order is $\Theta(\Psi_q(S))$ for every member of the class.*

Proof. Every balanced band has size at least $2s$. Nonterminal bands do so by construction; a terminal band is either at least its predecessor or merged with it. Therefore $s_r \leq s \leq m_r/2$ on each occupied band and

$$\log|\mathcal{C}(\mathbf{s})| = \sum_r \log\binom{m_r}{s_r} = \Theta\left(\sum_{r:s_r>0} s_r \log(em_r/s_r)\right) = \Theta(\Phi(\mathbf{s})).$$

The last equivalence absorbs the additive one because $s \geq 2$. The same bounded-error transcript argument as in Appendix A.6 gives the lower bound. Lemma 5 gives the matching rank-band upper bound and the equivalence with Ψ. □

Together with the design-weight reduction, the same counting identity also implies an existence guarantee from general candidate search. The proposition is used here as a structural interpretation of the functional, not as an independent new adaptation principle.

C.5 Average precision and tail location

Average precision is $\mathrm{AP}_q(S) = s^{-1}\sum_i P_i$. The arithmetic–geometric mean inequality gives

$$\Psi_q(S) \geq s(1 - \log \mathrm{AP}_q(S)). \tag{21}$$

There is no converse upper bound controlled by a near-one AP value alone. For the support with $\rho_i = i$ for $i < s$ and $\rho_s = d$,

$$\mathrm{AP}_q(S) = 1 - \frac{1}{s} + \frac{1}{d}, \qquad \Psi_q(S) = s + \log(d/s).$$

As $s \to \infty$ and $d/s \to \infty$, AP tends to one while the functional's additive tail penalty diverges.

A family version makes the associated search cost explicit. Fix the first $s-1$ positives and let the last positive vary over a known set J of $L \geq 2$ ranks larger than $s-1$. Every member has $\mathrm{AP} \geq 1-1/s$, yet bounded-error transcript counting gives $\Omega(\log L)$ queries to recover the unknown tail position. Queries on subsets of J implement binary search, so the class minimax scale is $\Theta(\log L)$. The lower bound concerns locating an unknown member of this family; a predesignated member can still be verified by its complement.

D Related Work and Connections to General Search

D.1 Ranking quality and prediction-assisted search

AUC has a pairwise interpretation in ROC analysis (Fawcett, 2006). Bipartite-ranking theory studies the statistical risk of scorers and its relationship to class-probability estimation (Menon and Williamson, 2016). Our input is a fixed total order on a finite pool. The empirical AUC of that order defines a candidate class, and the quantity to be optimized is the number of subsequent OR queries. Thus the characterization concerns the recovery value of a ranking after it has been produced; it does not require a particular training loss or a probabilistic model of prediction errors.

Algorithms with predictions seek improved performance when advice is accurate while controlling the cost of inaccurate advice. Foundational examples include caching (Lykouris and Vassilvitskii, 2018), ski rental, and scheduling (Purohit et al., 2018). Closer to our query setting, Bai and Coester (2023) relate the comparison cost of sorting to errors in predicted positions or comparisons. Dinitz et al. (2024) study searching a sorted array with a predicted distribution over target keys, relating expected query cost to entropy and distributional error. These tasks recover an order or a target key through comparisons. Here an arbitrary subset query returns one OR bit, the unknown object is a size-s support, and the benchmark is a uniform budget on an exact-AUC class. The central connection is the use of prediction quality to quantify query cost; the class geometry and permitted queries determine the resulting rate.

Recent search data structures further exploit predicted access frequencies. Fu et al. (2025) develop learning-augmented skip lists and KD trees, while Chen et al. (2025) construct search trees with priorities derived from predicted weights and support arbitrary access distributions. Their guarantees connect predicted frequencies to search depth and expected access cost. Our ordinal input orders the locations of an unknown positive support, and the exact-AUC curve quantifies the cost of recovering that entire support. A complementary direction explicitly integrates predictor learning into the algorithmic task (Eliáš et al., 2024). The quality requirement in Corollary 1 supplies a target for a ranker used by a recovery procedure.

D.2 Side information, candidate classes, and quality benchmarks

Classical adaptive group testing identifies an unknown support through OR queries; generalized binary splitting exploits the known cardinality (Dorfman, 1943; Hwang, 1972). For a broader account of algorithms, converses, and model variants, see Aldridge et al. (2026). Side information can further restrict or weight the candidate supports. In prior-statistics group testing, itemwise defect probabilities guide test design (Li et al., 2014). Hypergraph models allow a family of possible supports (Gonen et al., 2022) and a distribution over that family to represent correlations (Nikpey et al., 2024). In a recent non-adaptive setting, Ravi and Shomorony (2025) use a Markov model for correlated infections and quantify the test savings from longer infection bursts. Our benchmark specifies the family through one empirical ranking statistic and asks for a uniform worst-case budget with per-support success probability at least $2/3$.

An exact-AUC class is itself a known hypergraph whose edges have size s. Consequently, general candidate-search tools apply: the weighted guarantee of Triesch (1996) yields the upper bound used in Lemma 3. The additional task is to characterize the OR-query complexity of these particular hypergraphs for every prescribed rank sum. The distinction matters near the boundary, where their logarithmic cardinality is $\Theta(\sqrt{D})$ but their query budget is $\Theta(D)$. The query-imbalance phenomenon has precedents in general set-system search (Gonen et al., 2022); our constructions enforce it under each exact quality constraint and connect it to the plateau and logarithmic regimes.

The role of query geometry also appears in active hypothesis identification. Nowak (2011) gives conditions under which generalized binary search identifies one of N hypotheses in $O(\log N)$ queries. Our small-displacement classes illustrate why candidate counting alone cannot guarantee such a rate for OR queries. Generalized binary search in this comparison refers to hypothesis splitting; the experimental GBS baseline is the distinct generalized binary splitting procedure of group testing.

Adaptation also has a general-search interpretation. Competitive group testing compares an algorithm that does not know the defective count with an optimum that does (Du and Hwang, 1993). Our comparison instead holds s fixed and withholds D. The following two subsections make the connections explicit: rank-based design weights recover the ordinal upper-bound scale, and verification over nested quality classes gives an alternative existence proof of adaptation. The window algorithm in the main text implements the exact-AUC rates directly, without maintaining a candidate hypergraph.

D.3 Rank-only design weights

Set $w_j = \min\{1, (s/j)^2\}$ in rank coordinates and define

$$p_q(S) = Z_s^{-1} \prod_{j \in S} w_j, \qquad Z_s = \sum_{|A|=s} \prod_{j \in A} w_j.$$

The tail integral gives $\sum_j w_j \le s + s^2 \sum_{j>s} j^{-2} \le 2s$, hence $Z_s \le (2s)^s/s! \le (2e)^s$. Therefore

$$\begin{aligned} \log \frac{1}{p_q(S)} &= \log Z_s + 2 \sum_{i=1}^{s} \log \max\{\rho_i/s, 1\} \\ &\le s \log(2e) + 2 \sum_{i=1}^{s} \log(\rho_i/i) = O(\Psi_q(S)). \end{aligned}$$

Lemma 3 gives an $O(\Psi)$ algorithm using only q, s. This reduction explains why a worst-case ordinal input model does not, by itself, separate the existence of such an upper bound from weighted search. A design distribution specifies an algorithm; it need not describe how the true labels were generated.

D.4 Verification and adaptation over nested classes

The explicit procedure in Section 4 admits a general comparison with quality-aware candidate search. For $0 \le B \le sh$, let $\mathcal{C}_{\le B} = \bigcup_{j=0}^{B} \mathcal{C}_j$. Lemma 1 and the counting upper bound equation 8 provide deterministic class-aware procedures with known nondecreasing budgets

$$L(B) \le C(1 + F_s(B))$$

for a universal constant C; increase C if needed to use the expression on the right as an explicit budget. At $B = 0$, the unique support is known.

Consider a size-s proposed support $\widehat{S}$. A negative answer on $[d] \setminus \widehat{S}$ implies $S \subseteq \widehat{S}$, and equality of cardinalities then certifies $S = \widehat{S}$. This certificate remains valid even if the algorithm that proposed $\widehat{S}$ was run under a false quality assumption.

Run stages with query budgets $1, 2, 4, \ldots$. At a stage select the largest B whose stated class-aware budget fits, or skip the stage if none fits. Run that class-aware procedure with a strict cutoff. If answers are inconsistent with its candidate class, or it gives no size-s output, discard the stage. Otherwise verify the proposed support with its complement query and accept only after a negative answer.

At the first stage of budget at least $C(1 + F_s(D_q(S)))$, the chosen class contains the true support, so the procedure finishes correctly and its verification succeeds. The sum of all preceding query budgets, including at most one verification per stage, is a constant multiple of this budget. This gives $O(1 + F_s(D_q(S)))$ without knowing $D_q(S)$. Interleaving the two ranking directions gives $O(1 + F_s(\Delta_q(S)))$.

Competitive group testing traditionally compares unknown-cardinality procedures with a cardinality-aware optimum (Du and Hwang, 1993); here the cardinality is known and the quality statistic is unknown. The reduction above explains the existence of adaptation. The window algorithm implements it directly without enumerating the nested classes and additionally preserves the fine-grained bound of Proposition 2.

E Finite-Class Optimization and Certificates

E.1 Exact deterministic search

Fix an exact-AUC class $\mathcal{C}$ of n candidates and let $h(A)$ be the minimum zero-error deterministic depth when the true support is promised to lie in $A \subseteq \mathcal{C}$. For $|A| \le 1$, set $h(A) = 0$. Otherwise, writing $A_Q^- = \{S \in A : S \cap Q = \varnothing\}$ and $A_Q^+ = A \setminus A_Q^-$, the Bellman recurrence is

$$h(A) = 1 + \min_{Q : A_Q^- \neq \varnothing,\ A_Q^+ \neq \varnothing} \max\{h(A_Q^-), h(A_Q^+)\}. \tag{22}$$

The solver enumerates all $2^d - 1$ nonempty pools. Pools with the same candidate partition may be merged. An uninformative pool can be omitted from a minimum-depth tree, so the recurrence covers every adaptive OR strategy. All proper subsets are computed before A, using integer arithmetic, and an optimal pool witness is retained at every nonsingleton state. The root value is $M_{\mathrm{det}} = h(\mathcal{C})$.

E.2 Randomized budgets as coverage by successful subsets

Fix a budget t and define $\mathcal{A}_t = \{A \subseteq \mathcal{C} : h(A) \le t\}$. A deterministic depth-$t$ procedure that is correct on a subset A restricts to a zero-error tree on A, so $A \in \mathcal{A}_t$. Conversely, an optimal tree for $A \in \mathcal{A}_t$ extends to all of $\mathcal{C}$ by arbitrary outputs on branches inconsistent with A; it has depth at most t and is correct on every member of A.

Consequently a randomized budget-t procedure with per-instance success at least $2/3$ exists exactly when nonnegative weights satisfy

$$\sum_{A \in \mathcal{A}_t} w_A = 1, \qquad \sum_{\substack{A \in \mathcal{A}_t \\ S \in A}} w_A \ge \frac{2}{3} \quad \text{for every } S \in \mathcal{C}. \tag{23}$$

A mixture of the corresponding trees realizes these weights. It suffices to consider inclusion-maximal members of $\mathcal{A}_t$, since enlarging a successful subset can only improve coverage. For infeasibility, a distribution π on $\mathcal{C}$ with $\max_{A \in \mathcal{A}_t} \pi(A) < 2/3$ certifies that no mixture can meet the per-instance requirement. Averaging the success probability over π proves this claim. Thus the calculation solves the uniform-budget randomized problem itself, rather than an entropy relaxation.

E.3 Certificates and coverage

Linear programming proposes mixture weights or an adversarial distribution. These are converted to normalized rational numbers and checked with exact fraction arithmetic. A floating-point feasibility tolerance alone never determines a reported budget. A separate checker reconstructs all OR partitions, verifies every Bellman minimum and its witness, tests every rational coverage inequality, and verifies the recovery trees for all positive-weight mixture components. For lower certificates, it checks the bound against every feasible candidate subset.

Table 2: Coverage of the exact finite-class calculation. Support records count members separately across different (d, s) problems. The final column counts classes in which randomization reduces the budget relative to zero-error deterministic recovery. All certificates pass verification.

d	Classes	Support records	Largest class	$M < M_{\text{det}}$
4	5	6	2	0
5	7	10	2	0
6	19	35	3	5
7	24	56	5	6
8	46	154	8	18
9	55	246	12	21
Total	156	507	12	50

The accompanying code includes the complete class table and certificates. Its independent checker verifies the frozen certificates without invoking the optimization solver. The largest class has only 12 supports, making enumeration over candidate subsets feasible. This finite solver is used to study the benchmark, not as an implementation of AUC-oblivious recovery.

E.4 Finite overhead of the explicit algorithm

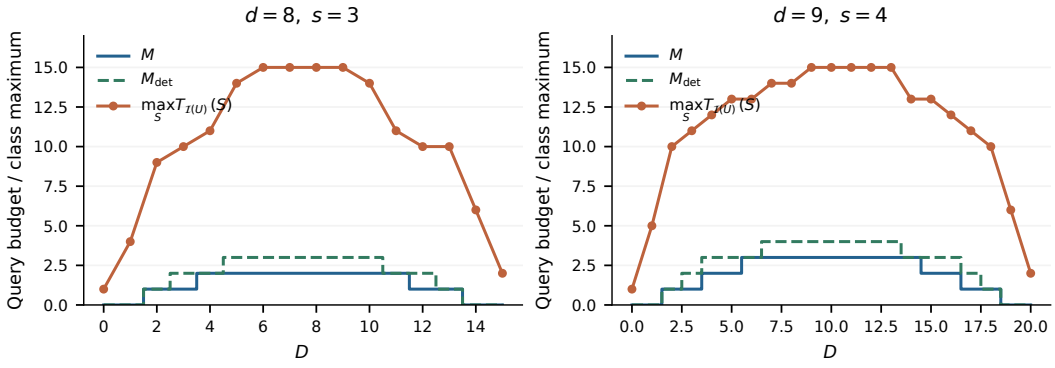


Figure 4: Exact finite-class budgets, together with $\max_{S \in \mathcal{C}_D} T_{\mathcal{I}(U)}(S)$ from complete enumeration. The optimizer knows D and $\mathcal{I}(U)$ does not. The latter curve is a class maximum of one fixed implementation.

At $(d, s, D) = (9, 4, 10)$, the maximum cost of $\mathcal{I}(U)$ is 15, compared with $M = 3$ and $M_{\text{det}} = 4$. The singleton benchmarks are zero even though the oblivious algorithm must spend queries to certify its answer. These finite differences illustrate the constants and additive term in Theorem 2. Reversal

symmetry holds for the optimized budgets; a fixed forward-first schedule need not have exactly symmetric realized costs.

F Synthetic Protocols and Additional Illustrations

F.1 Exact-displacement generation

Let $N(k, h, D)$ count partitions of D into at most k parts, each at most h. The recurrence

$$N(k, h, D) = N(k, h-1, D) + N(k-1, h, D-h)$$

separates partitions without a part equal to h from those with one such part removed. Use $N(k, h, 0) = 1$ for nonnegative k, h, and zero for infeasible arguments. Then $N(s, d-s, D) = |\mathcal{C}_D|$. Recursive unranking chooses a branch by its exact integer count, reconstructs a partition, pads it with zeros, and forms $\rho_i = i + e_i$. Uniformly sampled integer indices therefore give uniform support draws.

For each of $(96, 3), (160, 6), (224, 12)$, use the distinct feasible values in

$$\{0, 1, 2, \lfloor s/2 \rfloor, s, 2s, \lfloor s^2/2 \rfloor, s^2, 2s^2, 4s^2, \lfloor s(d-s)/2 \rfloor\}.$$

Classes of size at most 48 are completely enumerated; otherwise sample 48 indices independently with replacement. The support-generation seed is 2026091401. Each base support is reflected to the opposite orientation, together with its GBS permutations. When $2D = s(d-s)$ the central class is recorded once. The resulting uniform-class evidence comprises 62 class-orientation settings and 1,544 support-draw records. Repeated draws remain separate records with their original weights.

F.2 Algorithms, aggregation, and uncertainty

All five primary algorithms know s and use the same sound-inference wrapper: random-order GBS, ranked GBS, ranked singleton scanning, rank-band recovery, and U. Only random-order GBS receives an order independent of the prediction. Ranked GBS uses the identical procedure with the input order set to the predicted ranking. The GBS implementation uses cardinality-aware group selection and bisection within known-positive groups, with eight independent random orders per base support. The scheduler for U alternates logical requests 1:1, forward first, including requests answered by inference. No AUC or class identity is passed to an evaluated algorithm.

For each support record, first average GBS costs over its eight orders. Class means and plotted ratios are then

$$\overline{T}_A = \frac{1}{n}\sum_{j=1}^{n} T_A(S_j), \qquad R_A = \frac{\overline{T}_A}{\overline{T}_{\mathrm{GBS}}}, \quad A \in \{\mathcal{I}(U), \text{ranked GBS}\}.$$

Thus each R_A is a ratio of means, rather than the mean of per-support ratios. For sampled classes, Figure 2 reports percentile intervals from 2,000 paired bootstrap resamples of the existing support records, keeping each record's algorithm costs together. Seeds are $2026091402 + 100000d + 1000s + \Delta$; mirrored settings use the same resampling indices. Intervals describe support-sampling variability conditional on the recorded GBS-order averages. No bootstrap interval is attached to a fully enumerated class. Connecting segments guide the eye between evaluated displacements. These are empirical means, not estimates of a class maximum.

All recorded runs recover the exact support. Random-order GBS gives the reference cost without ordinal information; ranked GBS isolates the effect of supplying that information to the same recovery procedure.

F.3 Prescribed-sum mechanisms across scales

We evaluate the proof-defined small-displacement and shifted-block peeling families by complete enumeration. For the former, $d = 2s$ and $D = s$ with $s \in \{4, 8, 16, 32, 64\}$; for the latter, $d = 4s$ and $D = s^2/2$ with $s \in \{8, 16, 32, 64\}$. For the counting regime, fix $s = 8$, set $D = \alpha s^2$ and $d = (2\alpha + 1)s$, and take $\alpha \in \{4, 8, 16, 32, 64\}$. The construction protocol uses seed 20260924. At each counting setting we draw 48 supports uniformly with replacement from the exact-sum symmetric-pair family. The plots distinguish measured costs from analytic lower bounds: $\lceil 2m/3 \rceil - 1$ for a

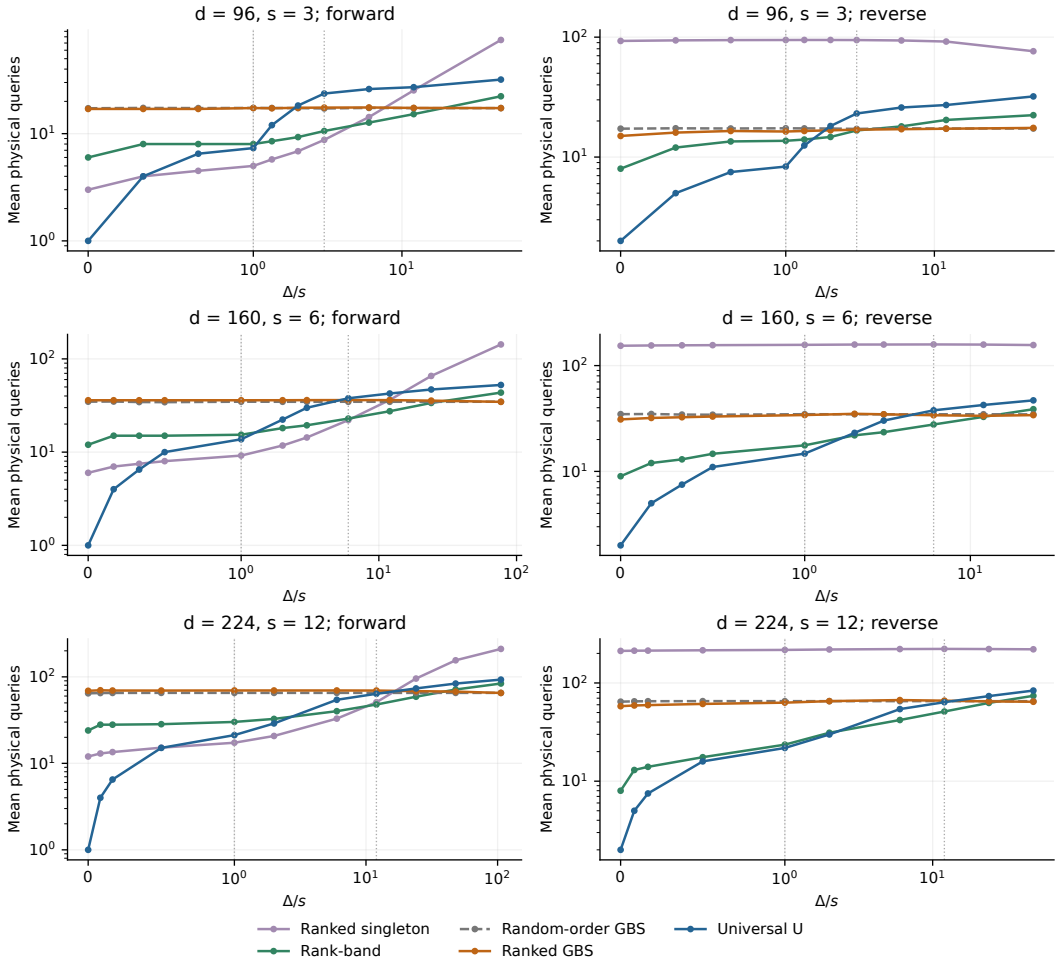


Figure 5: Complete primary-baseline comparison for the frozen uniform-class protocol. Every algorithm uses the same inference rules. Rows specify (d, s) and columns specify orientation; the vertical axis is logarithmic. A self-reflecting central class is recorded only in the forward panel. Dotted vertical lines mark $\Delta = s, s^2$.

peeling family of size m, and $\log_2(2|\mathcal{H}|/3)$ for a counting family $\mathcal{H}$. Observed maxima are full subfamily maxima for enumeration and sample maxima for the symmetric-pair runs.

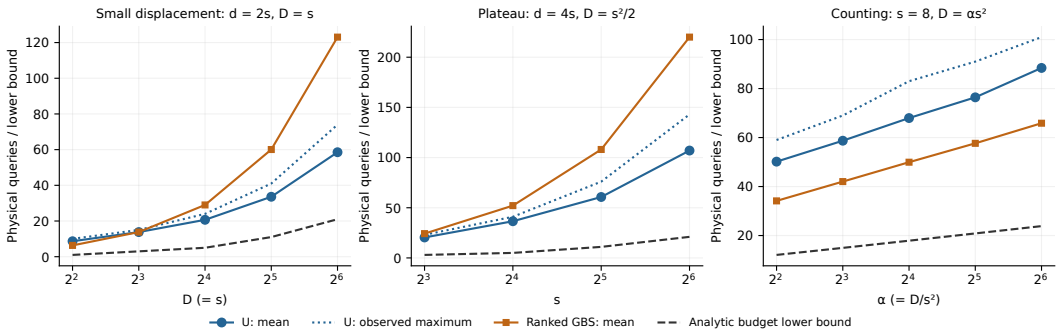


Figure 6: Proof-defined families across the three regimes. Solid curves show measured mean costs; the dotted curve shows the observed maximum of $\mathcal{I}(U)$. Peeling families are enumerated completely, whereas the counting panel uses 48 draws at each size. Dashed black curves give analytic lower bounds on the uniform recovery budget.

F.4 Constructed separation within one exact-AUC class

For $s \in \{4, 6, 8, 10, 12, 16\}$, let $d = 2s^3$ and $I = \{s^2 + 1, \ldots, 2s^2\}$, as in the separation proof. Exact integer dynamic programming selects a largest rank-sum fiber of size-s subsets of I, breaking ties by the smaller sum. We sample 48 supports uniformly with replacement from this fiber and compare them with $\{1, \ldots, s-1, s+D_0\}$ at the identical displacement D_0. Fiber cardinalities are checked independently by partition counting. The mean $\mathcal{I}(U)$ cost on the sampled hard fiber rises from 30.96 to 136.35 over the evaluated sizes, while the constructed easy support's cost rises from 27 to 70. These are finite illustrations of the shared-AUC construction; the analytic counting bound applies to the entire hard fiber, including when that fiber is supplied to the algorithm. At $s = 16$, the fiber has 13,888,244,935,445,960,871,352 supports. The integer counting bound requires a uniform budget of at least 73 queries for success probability $2/3$ on every member, while the same $\mathcal{I}(U)$ implementation recovers the constructed easy support in 70 queries. Both have $(d, s, D) = (8192, 16, 6016)$.

Table 3: Exact fibers used in Figure 3. The integer lower bound is $\lceil \log_2(2|\mathcal{H}|/3) \rceil$. Easy and hard columns report $\mathcal{I}(U)$ costs; the latter is a mean of 48 uniform draws.

s	D_0	AUC	Budget lower bound	Easy cost	Hard mean
4	88	0.8226	6	27	30.96
6	306	0.8803	15	33	48.42
8	736	0.9094	25	40	64.40
10	1450	0.9271	36	46	83.58
12	2520	0.9390	48	54	100.10
16	6016	0.9540	73	70	136.35

F.5 Complete classes and geometric descriptors

The two complete classes $(d, s, D) = (96, 3, 9)$ and $(96, 3, 18)$ contain 12 and 37 supports. Their $\mathcal{I}(U)$ costs range over 10–26 and 20–30, respectively. Figure 8 plots these costs against Ψ, with no fitted regression line.

For example, $\{1, 2, 12\}$ has $\Psi \approx 4.386$ and costs 25 queries, whereas $\{4, 5, 6\}$ has $\Psi \approx 5.996$ and costs 10. An upper-bound functional need not order exact algorithm costs.

The supplied records also include the comparator $\log(e\rho_s/s)$, the deepest-positive log-rank scale. Together these descriptors summarize rank geometry at different resolutions; exact costs also depend on the splitting and verification paths.

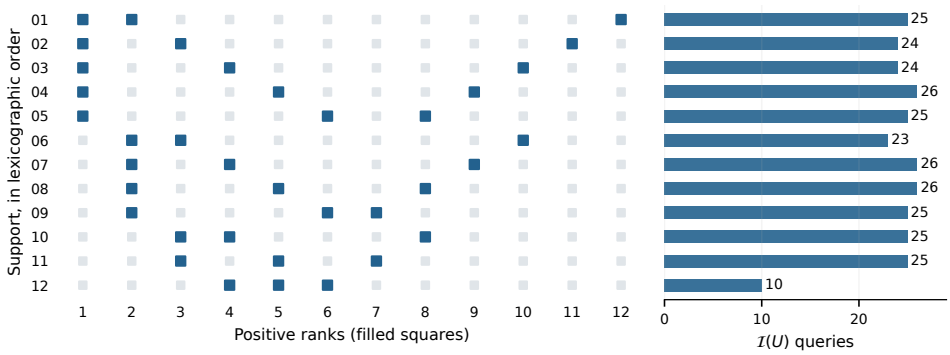


Figure 7: Every support of $(d, s, D) = (96, 3, 9)$, in lexicographic order. Filled squares mark positive ranks; all omitted ranks are negative. The right panel records the cost of $\mathcal{I}(U)$ on each support.

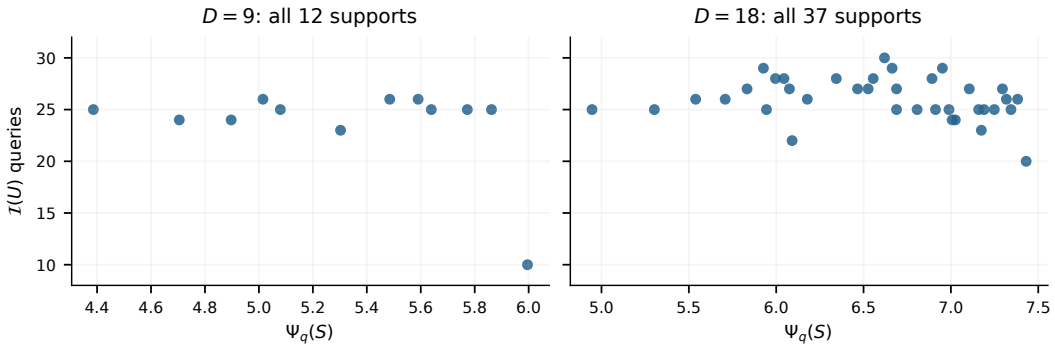


Figure 8: All members of two exact-AUC classes, with their positive-rank functional and fixed-algorithm cost. Neither panel asserts a monotone cost relationship. The $D = 9$ panel contains precisely the supports in Figure 7.

F.6 An analytic illustration of the entropy gap

Set $s = D$ and $d = 2D$. The exact class has $p(D)$ members, while the negative-pair subfamily contains $m = \lfloor (D + 2)/2 \rfloor$ candidates. Its exact randomized budget is

$$t_{\text{peel}} = \lceil 2m/3 \rceil - 1.$$

The lower bound is Lemma 4. For attainment, choose $t + 1$ candidates uniformly without replacement, test one position from the negative pair of each of the first t, and output the final selected candidate if all these tests are positive. A selected true candidate is always recovered, so every candidate succeeds with probability $(t + 1)/m \geq 2/3$.

At $D = 2048$, the class entropy is about 153.67 bits and the subfamily budget is 683 queries. This illustrates the small-displacement obstruction with explicit finite numbers; the all-parameter statement is established by the proof in Appendix A.

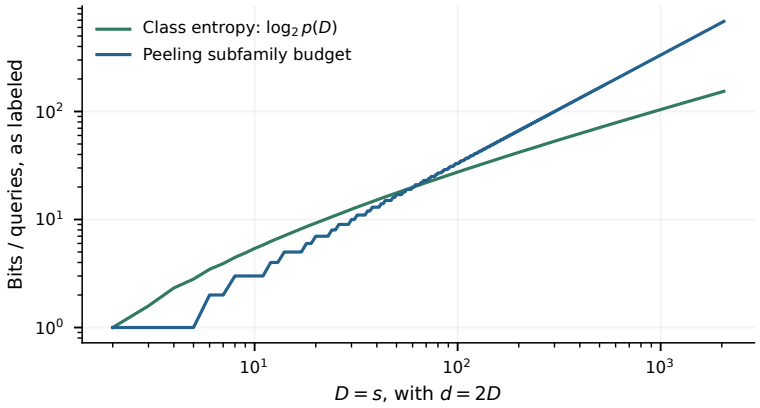


Figure 9: Computed class entropy and a construction-based OR-query lower bound. The partition numbers are exact integers; the plotted entropy is $\log_2 p(D)$. The subfamily budget is also a lower bound for the full exact-AUC class. These quantities are combinatorial calculations, not measured algorithm runtimes.

G Learned Ranking Quality Relative to Sparsity

G.1 Ranking models and held-out pools

We evaluate learned rankings for HIV and PCBA (Wu et al., 2018). The models are logistic regressions on radius-2, 1,024-bit Morgan fingerprints (Rogers and Hahn, 2010), with inverse regularization $C = 1$, balanced class weights, the `liblinear` solver, a 2,000-iteration limit, and seed 2027. HIV uses an 80/10/10 scaffold split (Bemis and Murcko, 1996); PCBA uses a shared 80/10/10 random row split across assays. All nine fitted models converged. Task-specific missing labels are excluded.

The eight PCBA assays are selected by label support, not ranking performance: at least 100 training positives, 20 validation positives, 40 test positives, 2,000 training negatives, and 1,016 test negatives. Among eligible assays, selection orders by decreasing minimum positive count across the three splits, then decreasing labeled test count, then task identifier, and keeps the first eight. The split assignments and selected tasks are retained with the experiment records. Full-precision decision scores determine the order within each sampled pool. Selected items are first shuffled with the sampling generator; score ties are then broken by this shuffled pool index.

We form 3,000 test pools per setting by sampling exactly s positives and $d - s$ negatives without replacement within each pool. HIV uses $(256, 4), (512, 8), (1024, 8), (1024, 16)$; each PCBA assay uses $(1024, 8)$. This yields 36,000 held-out pools. Pool membership may overlap across replicates. The five algorithms and inference policy match the synthetic comparison; random-order GBS again averages eight orders per pool. The original 396,000 runs and 36,000 additional ranked-GBS runs all recover their support exactly. Queries simulate OR responses on recorded labels.

G.2 Quality calibration and recovery cost

The theory gives a sparsity-dependent scale for interpreting learned AUC. Define the normalized displacement

$$\eta = \frac{\Delta}{s^2} = \frac{d - s}{s} \min\{\mathrm{AUC}, 1 - \mathrm{AUC}\}.$$

Corollary 1 identifies $\eta = O(1)$ as the quality scale for an $O(s)$ worst-case class budget; $\eta = 1$ locates the transition to the logarithmic regime. For the PCBA setting $(d, s) = (1024, 8)$, this transition on the high-AUC side occurs at $\mathrm{AUC} = 1 - 8/1016 \approx 0.9921$. Assays 2551 and 588342 both achieve mean AUC near 0.889, yet their mean normalized displacements are 14.08 and 14.06, respectively. These values are computed pool by pool before averaging. Across all 36,000 pools, 99.925% have $\eta \geq 1$. Thus an AUC close to 0.9 can coexist with displacement well beyond the

scale $\Delta = s^2$ when positives are sparse. The value $\eta = 1$ marks a scale in the minimax expression; finite comparisons between algorithms also depend on their constants and recovery paths.

Table 4: Held-out learned-ranking case study: mean queries over 3,000 pools per setting. AUC is the mean within-pool AUC. Numeric task identifiers denote PCBA assays. All algorithms use inference; GBS denotes the random-order baseline. Ranked GBS receives the same order as the other ranked procedures.

Task	(d, s)	AUC	GBS	Ranked GBS	Singleton	Rank-band	$\mathcal{I}(U)$
HIV	$(256, 4)$	0.707	27.65	27.76	165.17	30.22	40.77
HIV	$(512, 8)$	0.715	57.17	58.07	403.22	64.00	75.22
HIV	$(1024, 8)$	0.715	65.21	66.11	805.53	73.77	85.92
HIV	$(1024, 16)$	0.712	116.67	118.92	917.91	133.33	145.25
1030	$(1024, 8)$	0.755	65.20	66.31	683.56	74.14	87.67
2551	$(1024, 8)$	0.889	65.20	66.78	441.50	58.49	74.16
504332	$(1024, 8)$	0.789	65.21	66.45	620.86	71.27	85.64
504333	$(1024, 8)$	0.840	65.20	66.64	550.49	66.26	80.99
504339	$(1024, 8)$	0.827	65.20	66.58	586.68	67.19	81.61
588342	$(1024, 8)$	0.889	65.20	66.78	398.00	60.70	76.71
686978	$(1024, 8)$	0.821	65.21	66.59	554.71	69.16	83.92
686979	$(1024, 8)$	0.803	65.22	66.51	586.63	70.75	85.26

The corresponding query traces show that forward boundary recovery issues fallback queries on 99.7%–100% of pools in each setting: boundary certification rarely completes recovery directly. The current $\mathcal{I}(U)$ implementation exceeds GBS mean cost in all 12 settings, by 13.7%–47.5%. Ranked GBS exceeds random-order GBS by 0.41%–2.43%, and costs less than $\mathcal{I}(U)$ in every setting. Standalone rank-band recovery improves on both GBS variants on two PCBA assays. Section G.3 locates the additional costs that consume these savings. Together, the quality calibration and the paired ablation distinguish the ranking's position on the class curve from the finite cost of attaining that curve with the explicit algorithm.

The table reports descriptive means over the fixed pools, which may share molecules across replicates and assays. The anonymous code provides separate entry points for rebuilding the figures, recomputing finite and synthetic results, replaying all held-out ranked pools, and retraining the ranking models from public data. Training manifests and frozen ranks retain the settings needed for reproduction.

G.3 Where the additional queries arise

We replay three procedures on each of the same 36,000 ranked pools: standalone rank-band recovery, the forward procedure $B(q, s)$, and U. All use the same inference wrapper. Write their physical query costs as R, B, and U, and let G be the pool's mean over the eight recorded GBS orders. The paired decomposition

$$U - G = (R - G) + (B - R) + (U - B) \tag{24}$$

separates the ranked-search comparison, the net cost of adding boundary exploration, and the net cost of adding bidirectional scheduling. These are sequential algorithm ablations: shared facts and early termination are included in each difference. In particular, $U - B$ measures the net scheduling effect, whereas simply counting reverse queries would also count queries that help forward recovery. All 108,000 replayed runs recover exactly; the 72,000 rank-band and U costs match the original records individually.

Across the 12 settings, adding the forward boundary procedure costs 3.93–6.00 queries on average; adding bidirectional scheduling costs a further 5.92–11.01. Physical stage accounting makes this overhead concrete. On every PCBA pool, each direction issues one initial verification query and four suffix queries, with no window singleton queries. Thus U spends exactly ten queries on boundary probes. Its remaining, jointly informed rank-band calls exceed standalone R by 3.53–6.01 queries on average across these eight tasks. The query-stage records and the paired differences are both included in the reproducibility archive.

Table 5: Paired cost decomposition on the fixed held-out pools. Bars denote means over the same 3,000 pools per row. The first three difference columns sum to the fourth before rounding. $\bar{B}$ reports the standalone forward procedure's mean cost. Positive differences indicate additional queries.

Task	(d, s)	$\bar{R} - \bar{G}$	$\bar{B} - \bar{R}$	$\bar{U} - \bar{B}$	$\bar{U} - \bar{G}$	$\bar{B}$
HIV	$(1024, 16)$	+16.66	+6.00	+5.92	+28.58	139.33
HIV	$(1024, 8)$	+8.56	+5.00	+7.15	+20.71	78.77
HIV	$(256, 4)$	+2.57	+3.93	+6.62	+13.12	34.15
HIV	$(512, 8)$	+6.83	+5.00	+6.22	+18.05	69.00
1030	$(1024, 8)$	+8.94	+5.00	+8.53	+22.47	79.14
2551	$(1024, 8)$	-6.71	+5.00	+10.67	+8.96	63.49
504332	$(1024, 8)$	+6.06	+5.00	+9.37	+20.43	76.27
504333	$(1024, 8)$	+1.06	+5.00	+9.72	+15.79	71.26
504339	$(1024, 8)$	+2.00	+5.00	+9.41	+16.41	72.19
588342	$(1024, 8)$	-4.51	+5.00	+11.01	+11.51	65.70
686978	$(1024, 8)$	+3.95	+5.00	+9.76	+18.71	74.16
686979	$(1024, 8)$	+5.53	+5.00	+9.52	+20.04	75.75

The two tasks where rank-band improves on GBS isolate the practical effect. For assay 2551, $\bar{R} = 58.49$ and $\bar{G} = 65.20$; forward boundary recovery raises the cost to $\bar{B} = 63.49$, retaining a mean advantage. Bidirectional scheduling adds 10.67 queries, giving $\bar{U} = 74.16$. For assay 588342, $\bar{R} = 60.70$ versus $\bar{G} = 65.20$; the boundary procedure adds 5.00 queries and bidirectional scheduling adds 11.01, giving $\bar{U} = 76.71$. These comparisons show how adaptation overhead can consume savings from the ranking. On the other ten settings, standalone rank-band already exceeds GBS, so removing this overhead alone would not reverse every comparison.

G.4 A FIXED DIRECTIONAL SCHEDULING ABLATION

We also evaluate the fixed 3:1 schedule, which issues three forward logical requests followed by one reverse request. It reduces mean cost relative to the 1:1 schedule by 2.85%–6.72% across all 12 settings, but remains 7.23%–37.55% above random-order GBS. On the synthetic grid it increases mean cost in 20 of the 30 reverse-orientation settings. We retain 1:1 as the primary schedule and report 3:1 as a fixed ablation.

Guarantee of the fixed scheduling ablation. Let C_f and C_r be the standalone logical query counts of the two boundary procedures. The 3:1 schedule completes within

$$\min\{4\lceil C_f/3\rceil, 4C_r\} = O(\min\{C_f, C_r\})$$

logical requests, and sound inference can only reduce physical calls. It therefore retains the simultaneous order guarantee, with different constants for the two orientations. The ablation concerns finite costs of a fixed scheduler.

Table 6: Fixed 3:1 scheduling on the same held-out pools, compared with the primary 1:1 schedule and both GBS baselines. All 36,000 additional 3:1 runs recover exactly.

Task	(d, s)	GBS	Ranked GBS	Rank-band	$\mathcal{I}(U)$	$\mathcal{I}(U_{3:1})$
HIV	$(1024, 16)$	116.67	118.92	133.33	145.25	141.11
HIV	$(1024, 8)$	65.21	66.11	73.77	85.92	81.84
HIV	$(256, 4)$	27.65	27.76	30.22	40.77	38.03
HIV	$(512, 8)$	57.17	58.07	64.00	75.22	71.17
1030	$(1024, 8)$	65.20	66.31	74.14	87.67	83.78
2551	$(1024, 8)$	65.20	66.78	58.49	74.16	69.91
504332	$(1024, 8)$	65.21	66.45	71.27	85.64	81.25
504333	$(1024, 8)$	65.20	66.64	66.26	80.99	76.42
504339	$(1024, 8)$	65.20	66.58	67.19	81.61	77.18
588342	$(1024, 8)$	65.20	66.78	60.70	76.71	72.46
686978	$(1024, 8)$	65.21	66.59	69.16	83.92	79.46
686979	$(1024, 8)$	65.22	66.51	70.75	85.26	80.96

H SCOPE AND EXTENSIONS

Our characterization isolates how ordinal prediction quality governs adaptive exact recovery in the known-cardinality, noiseless OR model. Verification links the quality-aware benchmark to an algorithm that does not know quality. We first quantify the role of pool size in this link, then describe how other measurement settings change the problem.

H.1 THE COST OF VERIFICATION WITH BOUNDED POOLS

We prove Proposition 1. Write $h = d - s$ and $S_0 = [s]$. The deterministic optimum is $\min\{s, \lceil h/L \rceil\}$; the randomized argument gives the explicit bound

$$\mathbb{E}[T_A(S_0)] \geq \frac{1}{3} \min\left\{ s, \frac{h}{\min\{L, h\}} \right\}.$$

The expectation is over the algorithm's internal randomness, so this also lower-bounds any uniform query budget on S_0.

Proof. First consider the deterministic query path on S_0. If it uses fewer than s queries, some $i \in S_0$ never appears as the unique positive item in a query. If it also uses fewer than $\lceil h/L \rceil$ queries, some $j \notin S_0$ is absent from every negative query. The support $S_{ij} = S_0 \setminus \{i\} \cup \{j\}$ then gives the same answer to every query on this path: negative queries still contain no positive, and positive queries retain a positive other than i. Adaptivity consequently follows the same path and gives the same output on two different supports, proving the lower bound.

For the upper bound, either query each of the s items of S_0 individually, or partition its complement into $\lceil h/L \rceil$ pools and query them. All-positive singleton answers, or all-negative complement answers, certify S_0 using the known cardinality. Choose the cheaper procedure. If verification fails, recover the support by singleton tests. The resulting algorithm is valid on every support and attains the stated cost on S_0.

For the randomized bound, consider all sh supports S_{ij} and couple the runs on S_0 and S_{ij} using the same internal randomness. Until their first differing answer, the queries and stopping decisions coincide. Per-instance success implies

$$\Pr_{S_0}\{\widehat{S} = S_0\} \geq \tfrac{2}{3}, \qquad \Pr_{S_{ij}}\{\widehat{S} = S_0\} \leq \tfrac{1}{3}.$$

Thus the probability that the coupled query answers differ is at least $1/3$ for each pair (i, j).

Fix the internal randomness and examine the path on S_0. A query intersecting S_0 in at least two items distinguishes no single-swap neighbor. A query intersecting it in exactly one item distinguishes at most h neighbors. A negative query distinguishes $s|Q| \leq s \min\{L, h\}$ neighbors. The union of the neighbors distinguished along this path therefore has size at most

$$T_A(S_0) \max\{h, s \min\{L, h\}\}.$$

Taking expectations and summing the sh individual probability lower bounds gives

$$\frac{sh}{3} \leq \mathbb{E}[T_A(S_0)] \max\{h, s \min\{L, h\}\},$$

which proves the explicit randomized bound. For $L \leq h$, $\lceil h/L \rceil \leq 2h/L$; for $L > h$, $\lceil h/L \rceil = h/\min\{L, h\} = 1$. The bound therefore has the order stated in Proposition 1. If the expected stopping time is infinite, the bound holds directly. □

At $d = s^2$ and $L = s$, the deterministic and randomized lower bounds are $s - 1$ and $(s - 1)/3$, respectively. The exact-AUC-aware benchmark remains zero on this singleton class. Thus bounded pools can preclude the constant additive adaptation guarantee even at perfect ranking.

H.2 Further measurement and prediction settings

Known cardinality and verification. The support size s is available to both the recovery algorithm and the class-aware benchmark. This information enables a central operation: a negative query on the complement of a proposed size-s support certifies complete recovery. It also permits inference and stopping once all s positives have been identified. With unknown cardinality, the same negative answer only certifies containment in the proposed support; it does not identify which of its items are positive. Cardinality uncertainty therefore changes both the verification mechanism and the appropriate benchmark. Extending the characterization to an unknown s, or to an interval of possible cardinalities, requires accounting for this additional uncertainty.

Noiseless responses and exact recovery. Each query returns the correct OR of the labels in its pool. The algorithms consequently use negative answers to exclude items and positive answers to constrain the remaining support. Under noisy responses, these deductions require statistical evidence, and query complexity must also account for the noise model and the target recovery probability. The per-instance success probability in our randomized minimax benchmark concerns algorithmic randomness; the measurement oracle itself is noiseless. A noisy analogue would characterize the joint dependence on ranking quality, measurement reliability, and recovery confidence.

Pool size and measurement cost. Queries may contain arbitrary subsets, and every oracle call has unit cost regardless of pool size. This convention measures the number of measurements and allows large-pool operations such as complement verification. Applications may instead impose maximum pool sizes or costs that depend on pool size, composition, or preparation. Such constraints change the available tests or their accounting: Proposition 1 quantifies the resulting change at the perfect-ranking endpoint. Characterizing the full dependence on D and L, or on a specified measurement cost, would extend this endpoint separation across the quality curve.

Adaptivity and measurement rounds. The algorithms can choose each query after observing earlier answers. Boundary certification and recursive splitting both use this sequential feedback. The present complexity measure counts oracle calls; it does not optimize the number of adaptive rounds or elapsed time when measurements can run in parallel. Batched and non-adaptive recovery impose additional restrictions on query design. Their natural benchmarks would track ranking quality together with a round budget, or characterize the tradeoff between measurement count and sequential depth.

Given rankings and learned predictors. The theory takes a total order as input and measures its empirical AUC on the finite recovery pool. Its guarantees apply to that order regardless of how the scores were trained or calibrated. They do not include the cost of obtaining training labels or fitting the predictor. Nor does a validation-set AUC automatically certify the AUC of a new recovery pool. An end-to-end analysis would connect prediction generalization, including changes in the test distribution, to the quality-dependent query budget. The quality–sparsity relation provides a quantitative target for that connection.

Minimax rates and finite implementation costs. The exact-AUC characterization concerns worst-case class budgets up to universal constants. The explicit algorithm attains these rates simultaneously over quality levels; its finite mean cost also reflects boundary exploration, scheduling, and the opportunities for shared inference. Appendix G examines these effects on fixed learned rankings, and Appendix G.3 separates them through paired algorithm ablations. These results identify a concrete algorithmic direction: reduce the finite cost of quality-oblivious adaptation while preserving its simultaneous guarantees. More broadly, the distinction between class complexity and realized query cost motivates evaluating both the quality-dependent benchmark and the implementation that attains it.