NEARLY LINEAR ORACLE ALGORITHMS FOR SUBMODULAR MAXIMIZATION UNDER MATROID AND POLYMATROID CONSTRAINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We give nearly linear oracle algorithms for monotone submodular maximization under general matroid constraints and for monotone DR-submodular maximization over integral polymatroids. In both settings, our algorithms achieve an expected $(1-1/e-\epsilon)$ approximation with polynomial dependence on the accuracy parameter. For general matroids, we obtain nearly linear complexity with either rank-oracle or independence-oracle access by combining new low-cost rounding procedures with a nearly linear fractional optimization scheme. For integral polymatroids, we obtain nearly linear complexity in the number of coordinate types even when the numerical rank is exponentially large, with only doubly logarithmic dependence on the rank. Our construction uses capacity-scale separation, count-based fractional optimization, and efficient local rounding.

1 INTRODUCTION

Submodular optimization plays an important role in modern machine learning, with applications ranging from data subset selection to structured decision-making problems. Examples include curating high-quality pretraining and fine-tuning datasets for large language models (Ji et al., 2024; Kumari et al., 2024; Agarwal et al., 2024), diversified online recommendation systems (Hiranandani et al., 2020; Chen and Crawford, 2025), multi-agent optimization in robotics (Zhou and Tokekar, 2022; Xu and Tzoumas, 2024), and precise image attribution in computer vision (Chen et al., 2024). Submodular functions informally satisfy a diminishing returns property that is exhibited by many objective functions for such problems. Formally, let $f : 2^U \to \mathbb{R}$ be defined over subsets of a ground set U of size n. Then f is *submodular* if for all $A \subseteq B \subseteq U$ and $x \notin B$, we have $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$. Further, f is *monotone* if $f(Y) \geq f(X)$ for every $X \subseteq Y \subseteq U$.

A fundamental problem in constrained submodular optimization is maximizing a monotone submodular function subject to a matroid constraint. Matroids capture many practical feasibility requirements, including matching constraints in advertising (Babaioff et al., 2007; 2008), capacity constraints in recommendation systems (Wilhelm et al., 2018), and connectivity constraints in routing and spanning-tree problems (Xu and Rodrigues, 2015). Classical greedy algorithms achieve a $1/2$-approximation for general matroids (Fisher et al., 1978), while continuous greedy achieves the optimal $(1 - 1/e)$ guarantee (Calinescu et al., 2011). Subsequent work has focused on retaining this near-optimal approximation while reducing running time and oracle complexity (Badanidiyuru and Vondrák, 2014; Henzinger et al., 2023).

The standard approach to monotone submodular maximization under a matroid constraint first computes a fractional solution and then rounds it to an integral independent set. Both stages can contribute substantially to the oracle complexity. For a general matroid with rank r accessed through an independence oracle, the fast continuous-relaxation framework uses $\widetilde{O}(n\sqrt{r})$ value-oracle queries to f and $\widetilde{O}(n\sqrt{r})$ independence-oracle queries to compute the fractional solution, while the rounding stage requires an additional $\widetilde{O}(r^{3/2})$ independence-oracle queries (Buchbinder et al., 2017; Kobayashi and Terao, 2024). Since the matroid rank satisfies $r \leq n$ and may be as large as $\Theta(n)$, these bounds become $\widetilde{O}(n^{3/2})$ in the high-rank regime. With rank-oracle access, the fractional stage can already

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.

be implemented in nearly linear time, but the $\widetilde{O}(r^{3/2})$ rounding term remains. The scaling issue is even more pronounced for polymatroid constraints, which generalize matroid constraints by allowing integer-valued multiplicities. The total rank, unlike the rank of a matroid, is not bounded by the ground-set size $n = |E|$ and may be exponentially large. Consequently, even a dependence of $\log^3 r$ can be prohibitive: if $r = 2^{\Theta(n)}$, then $\log^3 r = \Theta(n^3)$. Although a feasible solution can be stored compactly using only its n coordinate counts, standard optimization procedures can still incur query complexity depending on the numerical rank r, for example through marginal-threshold ranges, binary searches over coordinate counts, and sampling guarantees over the units of an optimum.

Our contributions. Motivated by the gap between optimal approximation guarantees and the high oracle complexity of existing methods, we develop nearly linear oracle algorithms for monotone submodular maximization under general matroid constraints and for monotone DR-submodular maximization over integral polymatroids, using new rounding, preprocessing, and count-based optimization techniques. In particular, our contributions are:

- **Nearly linear algorithms for general matroids.** We give a $(1-1/e-\epsilon)$-approximation algorithm for monotone submodular maximization under an arbitrary matroid using nearly linear value and independence-oracle queries. The same nearly linear guarantee holds with rank-oracle access. For fixed accuracy, this improves the previous $\widetilde{O}(n\sqrt{r})$ dependence for general independence-oracle matroids to $\widetilde{O}(n)$.
- **Nearly linear rounding for general matroids.** Given a fractional solution represented by a short list of matroid bases, we give new rounding algorithms whose query complexity is nearly linear in the matroid rank. The rank-oracle and independence-oracle variants require no additional value queries and lose only an arbitrarily small multiplicative factor in expected objective value.
- **Nearly linear fractional optimization with independence oracles.** We give a nearly linear algorithm for constructing the fractional solution needed by the rounding procedure using only ordinary value queries and a general matroid independence oracle. Combined with our rounding algorithm, this yields the first nearly linear general-matroid algorithm with polynomial dependence on the approximation accuracy.
- **Nearly linear algorithms for integral polymatroids.** We give a $(1 - 1/e - \epsilon)$-approximation algorithm for monotone DR-submodular maximization over arbitrary integral polymatroids using nearly linear value and integer-feasibility queries. The dependence on the numerical rank is only doubly logarithmic, so the algorithm remains nearly linear in the number of coordinate types even when the rank is exponential in the dimension.

The target approximation factor is essentially best possible in the value-oracle model: improving upon $1 - 1/e$ by a constant requires exponentially many value queries (Nemhauser and Wolsey, 1978; Vondrák, 2013). For fixed accuracy, even a cardinality constraint requires $\Omega(n/\log^2 n)$ value queries (Peng and Rubinstein, 2025), so nearly linear dependence on n is necessary up to polylogarithmic factors. We also prove a complementary $\Omega(n)$ expected integer-feasibility-query lower bound for polymatroids of rank 2^n, even for a known linear objective. Our main algorithmic result is as follows.

Theorem 1.1 (Main result)**.** *For $0 < \varepsilon \leq 1/2$, the following algorithms always return feasible solutions with expected value at least $(1 - 1/e - \varepsilon)\mathrm{OPT}$.*

(i) For a general matroid, $n\,\mathrm{poly}(1/\varepsilon, \log(2n))$ expected value and independence queries suffice. With a rank oracle, value and rank queries suffice within the same form of bound.

(ii) For a general integral polymatroid on n coordinate types, the algorithm uses $O(n\varepsilon^{-9}\Xi_\varepsilon^3)$ value queries and $O(n\varepsilon^{-11}\Xi_\varepsilon^3)$ expected integer-feasibility queries, where $\Xi_\varepsilon = \log(2n/\varepsilon) + \log\log(r+3)$. In particular, both costs are $n\,\mathrm{poly}(1/\varepsilon, \log(2n))$ for $r = 2^{\Theta(n)}$.

The guarantees are proved in Theorems E.1, F.1, and I.1, together with the composition in Appendix H.6. For the matroid problem, the theorem therefore matches the known $\Omega(n/\log^2 n)$ value-query lower bound up to polylogarithmic factors for fixed accuracy. This comparison concerns the dependence on n; it does not establish optimal dependence on $1/\varepsilon$ or a matching lower bound for every rank regime (see Proposition B.1). For polymatroids, Proposition I.26 proves an $\Omega(n)$ expected feasibility-query lower bound at rank exactly 2^n, even for a known linear objective. The

General-matroid algorithm	Oracle bound	Accuracy dependence
Badanidiyuru and Vondrák (2014), independence	$\widetilde{O}(nr)$	Polynomial
Buchbinder et al. (2017), swap rounding	$\widetilde{O}(n\sqrt{r}+r^2)$	Polynomial
Kobayashi and Terao (2024), independence	$\widetilde{O}(n\sqrt{r})$	Polynomial
Kobayashi and Terao (2024), rank	$\widetilde{O}(n+r^{3/2})$	Polynomial
Buchbinder and Feldman (2025a), randomized	$\widetilde{O}(n+r\sqrt{n})$	Exponential
Buchbinder and Feldman (2025b), randomized	$\widetilde{O}(n+r^{3/2})$	Exponential
This work, either oracle	$\widetilde{O}(n)$	Polynomial

Table 1: Bounds for an integral $(1-1/e-\varepsilon)$ approximation. Accuracy factors are omitted from the middle column and classified in the last. The full comparison and source locations are in Appendix B.

construction hides one valuable feasible coordinate among the candidates, while an independent zero-value coordinate supplies the exponential rank. Thus the dependence on the dimension is also tight up to polylogarithmic factors in this exponential-rank regime. Optimal dependence on accuracy and numerical rank remains open.

1.1 Preliminaries

A set function $f : 2^V \to \mathbb{R}_+$ is *submodular* if $f(e \mid A) \geq f(e \mid B)$ whenever $A \subseteq B$ and $e \notin B$, where $f(e \mid A) = f(A+e) - f(A)$. A *matroid* $M = (V, \mathcal{I})$ is a nonempty downward-closed family with augmentation: if $I, J \in \mathcal{I}$ and $|I| < |J|$, some element of $J \setminus I$ can be added to I. Its maximal independent sets are *bases*, all of size r. An independence oracle answers whether $A \in \mathcal{I}$; a rank oracle returns $r_M(A) = \max_{I \subseteq A,\, I \in \mathcal{I}} |I|$. One rank query tests independence. Computing rank from independence can require scanning the queried set, so polynomial equivalence does not preserve nearly linear complexity. These are classical oracle models (Edmonds, 1971; Karp et al., 1988); their distinction within submodular maximization also predates Kobayashi and Terao (2024), for example in Balkanski et al. (2019).

An *integral polymatroid* is $P(\varrho) = \{x \in \mathbb{R}^n_+ : x(A) \leq \varrho(A)\ \forall A \subseteq [n]\}$, where ϱ is normalized, monotone, integer-valued, and submodular. Its numerical rank is $r = \varrho([n])$, which need not be at most n. We maximize a monotone lattice function $f : \mathbb{Z}^n_+ \to \mathbb{R}_+$ satisfying *diminishing returns* (DR): $f(x+\mathbf{1}_e) - f(x) \geq f(y+\mathbf{1}_e) - f(y)$ for $x \leq y$. The constraint oracle tests integer feasibility in P; it does not return the polymatroid rank function. Matroid polytopes are the special case with singleton capacities at most one. Full definitions and examples are in Appendix A.

1.2 Related Work

The continuous-greedy framework of Calinescu et al. (2011) established the optimal $(1-1/e)$ approximation for monotone submodular maximization under a matroid constraint, and Badanidiyuru and Vondrák (2014) substantially accelerated this approach. Building on this line of work, Buchbinder et al. (2017) introduced a preprocessing–sampling tradeoff that improved the complexity for general matroids. Nearly linear algorithms were later obtained by Ene and Nguyen (2019) for settings in which maximum-weight bases can be maintained efficiently, including graphic and partition matroids, and Henzinger et al. (2023) extended this approach to additional explicitly represented matroid classes. These methods rely on specialized data structures and therefore do not directly apply to an arbitrary matroid given only through an independence oracle. For general oracle matroids, Kobayashi and Terao (2024) reduced the cost of rounding using longer exchange cycles and showed how the fast fractional phase can be implemented efficiently with rank-oracle access. Selected bounds for obtaining integral solutions are summarized below.

More recent work has improved the dependence on the ground-set size through different algorithmic approaches. The local-search and extended-extension methods of Buchbinder and Feldman (2025a;b) achieve stronger bounds in several regimes, but evaluating their auxiliary objectives incurs exponential dependence on $1/\varepsilon$. In particular, the extended-extension framework already provides a nearly linear fractional algorithm at fixed accuracy; combined with our new rounder, this would yield a nearly linear integral algorithm in that regime. Our additional preprocessing machinery is what allows

us to retain nearly linear complexity while also obtaining polynomial dependence on $1/\varepsilon$. Recent Poisson-process algorithms (Ganz Rozenman et al., 2026; Kulik et al., 2026) instead maintain integral bases through randomly timed exchanges. These methods are algorithmically distinct from our rounding and count-smoothing techniques and do not yield the same nearly linear ordinary-oracle guarantee for arbitrary matroids. Rank-parameterized algorithms (Nematollahi et al., 2026) focus on a complementary regime in which the matroid rank is treated as a parameter.

For integer-lattice objectives, Soma and Yoshida (2018) developed a $(1 - 1/e - \varepsilon)$-approximation framework over integral polymatroids that already uses binary-search batching of coordinate increments, a continuous extension, and a reduction of rounding to a local matroid. Their stated complexity includes $\widetilde{O}(n^3\varepsilon^{-5}\log^2 r)$ work in the fractional stage together with an $O(n^8)$ exact-decomposition cost for rounding. Our algorithm builds on this general approach while removing the polynomial dependence on the numerical rank and the expensive exact decomposition. Capacity-scale separation and an integer baseline control the range of relevant marginals, while geometric count searches and aggregate concentration reduce the remaining dependence on the numerical rank to doubly logarithmic. Finally, approximate local covering replaces exact decomposition and yields nearly linear rounding. Our construction also uses established packing, exchange-search, and multiplicative-weights primitives; we state their guarantees explicitly and distinguish these published ingredients from their new roles in our algorithms.

2 Nearly Linear Rounding for General Matroids

The standard approach to monotone submodular maximization under a matroid constraint first computes a high-value fractional solution and then rounds it to a single independent set. Even when the fractional stage is efficient, rounding can remain a dominant source of oracle complexity. This section addresses that bottleneck by developing nearly linear rounding procedures for both rank- and independence-oracle access.

To obtain a high-value integral solution from a fractional one, rounding procedures convert a fractional solution represented as a convex combination of matroid bases, $x = \sum_{i=1}^{t} \beta_i \mathbf{1}_{B_i}$, into a single base. A standard approach merges the bases pairwise, using feasible exchanges to make two bases progressively agree until they coincide. Classical swap rounding chooses exchanges whose effects cancel in expectation, while the faster method of Kobayashi and Terao (2024) obtains this cancellation using longer exchange cycles.

Our key idea is to avoid requiring the exchanges to form a complete cycle. Instead, we use short open sequences of feasible exchanges. The effects of the intermediate exchanges still cancel, leaving only a small loss associated with the two endpoints, which we control directly. This makes the exchange sequence cheaper to find while preserving nearly all of the multilinear value $F(x) = \mathbb{E}[f(R_x)]$. The resulting rounders return a single base with expected value at least $(1 - \eta)F(x)$ without making value queries. Since the fractional algorithms used later produce only $t = O(1/\varepsilon)$ bases, it suffices to make each pairwise merge nearly linear in the matroid rank. The rank- and independence-oracle cases differ in how the exchange sequence is constructed and how its endpoint loss is controlled. Our main result is stated as follows.

Theorem 2.1 (Nearly linear rounding for general matroids)**.** *Let $x = \sum_{i=1}^{t} \beta_i \mathbf{1}_{B_i}$ be a convex combination of bases of a rank-r matroid, and let $0 < \eta < 1$. There are rounding algorithms that return a single base S satisfying $\mathbb{E}[f(S)] \geq (1 - \eta)F(x)$ and make no value queries.*

With rank-oracle access, the algorithm uses $O(rt^2\eta^{-1}\log^2(2r))$ rank queries. With independence-oracle access, it uses $O(r(t-1)^2\eta^{-1}\log(2r))$ expected independence queries.

In particular, when $t = O(1/\varepsilon)$ and $\eta = \Theta(\varepsilon)$, both rounding procedures have nearly linear dependence on the matroid rank.

2.1 Algorithm Description

With a rank oracle, we can construct the open exchange walk efficiently. Suppose the two bases currently being merged differ in d elements. We choose one of these disagreement elements uniformly at random and then build a sequence of L feasible one-sided exchanges between the two bases. Each step can be found using $O(\log r)$ rank queries by binary searching over candidate exchange elements.

After constructing the walk, we choose one of its L exchange steps uniformly and perform the corresponding update, which reduces the disagreement between the two bases by one.

The value analysis charges only the starting endpoint of the walk. When the bases differ in d elements, choosing that endpoint uniformly bounds its expected charge by a $1/d$ share of the current multilinear value. Summing these charges as d decreases gives a harmonic factor, so a logarithmic walk length suffices. The full analysis appears in Appendix E.

With only an independence oracle, the same walk cannot be generated in the same direction. A rank query can test whether some element in a candidate set extends a fixed independent set, even when the full union is too large to be independent. An independence query on that union reveals only that it is dependent. Instead, we use the exchange-search primitive of Kobayashi and Terao (2024, Lemma 2.1), which fixes an element to insert and searches for an element that can be deleted while preserving feasibility. This reverses the direction in which the exchange walk can be constructed. It makes its uncontrolled endpoint the potentially harmful one, so the uniform-start argument from the rank-oracle case no longer applies.

We address this with geometric stopping. Stopping at a geometric time discounts its loss and associates that loss with an element actually removed during the merge. Because removed disagreement elements never return, these charges can be bounded globally by a single deletion-only comparison trajectory. Choosing the stopping parameter to distribute the allowed loss across the $t-1$ merges yields expected query complexity $O(r(t-1)^2\eta^{-1}\log(2r))$, with no value queries, while preserving at least a $(1-\eta)$ fraction of the fractional value. The geometric identity and global removal-budget argument are proved in Appendix F.

3 Nearly Linear Preprocessing for Independence-Oracle Matroids

The rounding algorithms of the previous section remove the integral-solution bottleneck once a short fractional base representation is available. With only independence-oracle access, however, constructing that fractional solution efficiently remains challenging. This section develops a preprocessing procedure that selects a small feasible prefix and simplifies the residual problem before fractional optimization begins. Its purpose is to reduce the total range of the remaining marginal values, so that the later continuous-greedy phase can estimate residual gains using far fewer value queries. This feasible prefix S can then be supplied to a range-sensitive continuous-greedy algorithm on the contracted matroid M/S, such as the decreasing-threshold framework of Badanidiyuru and Vondrák (2014) with the sampling approach of Buchbinder et al. (2017). The preprocessing bound reduces the cost of estimating residual marginal gains, allowing the fractional stage to remain nearly linear.

Theorem 3.1 (Nearly linear preprocessing)**.** *Let $M=(V,\mathcal{I})$ be a matroid given by an independence oracle, and let f be a normalized monotone submodular function. Fix $0<\kappa<1/2$ and $0<\zeta<1/2$, and suppose that $\mathrm{OPT}\le\Gamma\le 3\mathrm{OPT}$. With a deterministic query cap, the preprocessing algorithm uses $O(n\,\mathrm{poly}(1/\kappa,\log(2n),\log(1/\zeta)))$ value and independence queries and returns an independent prefix S. Except with probability $O(\zeta)$, this prefix satisfies*

$$A_S := \max_{I\in\mathcal{I}(M/S)} \sum_{e\in I} f(e\mid S) \le \rho\Gamma, \tag{3.1}$$

where $\rho = O\big(\log^2(2n)\log(2r)/(1-\kappa)\big)$. Moreover, for an analytical comparison base $C\supseteq S$,

$$\mathbb{E}[f(S)] \ge (\mathrm{e}-1)\mathbb{E}[\mathrm{OPT}-f(C)] - O(\zeta\mathrm{OPT}).$$

Choosing $\kappa,\zeta=\Theta(\varepsilon)$ supplies the preprocessing guarantees used for the overall $(1-1/e-O(\varepsilon))$ approximation.

The preprocessing considers the residual objective $g(I)=f(S\cup I)-f(S)$ on the contracted matroid M/S. The bound in (3.1) controls the singleton sampling ranges along every feasible residual direction, allowing the later range-sensitive continuous-greedy phase to use nearly linear total sampling cost.

The main difficulty is obtaining this guarantee without repeatedly rescanning the same elements. At each marginal-value threshold, we reduce the rank of the elements whose current marginal remains

large. A structural alternative lets us either process several bases with large disjoint private subsets, so that each expensive scan certifies many new low-marginal elements, or recurse on a smaller contraction minor where the same rank-reduction task is cheaper. Applying this idea across multiple levels yields nearly linear preprocessing with independence-oracle access. The remainder of the section develops this procedure and proves the range and value-preservation guarantees needed by the fractional algorithm. The complete preprocessing algorithm, range guarantee, value analysis, and query bound are given in Appendix G.

3.1 Private subsets or a smaller problem

Fix a threshold θ and call an element *high* if its current marginal $f(e \mid S)$ is at least θ. At this threshold, our goal is not to identify and remove every high element. Instead, it is enough to reduce the rank of the high-element restriction: once this rank is small, no feasible residual solution can contain many high-marginal elements, even if many mutually dependent high elements remain.

To reduce this rank efficiently, we repeatedly examine candidate bases. A candidate basis has one of two useful outcomes. If it contains many high elements, we select one of them uniformly and add it to the prefix S. The value gained by this selection can be used to pay for the expected loss it causes in a comparison optimum. If the basis contains few high elements, then a scan certifies most of its elements as low. By diminishing returns, an element certified low remains low as S grows during the current threshold pass. The difficulty is that different candidate bases may expose the same low elements, so repeatedly scanning bases need not produce enough new information to justify their construction.

We overcome this difficulty using a structural consequence of Quanrud's matroid-packing algorithm (Quanrud, 2024, Section 3.1 and Theorems 3.3, 4.2). Lemma G.3 shows that, given a rank-m matroid on ground set X and an integer b, **PackOrMinor** uses $\widetilde{O}(|X| + bm)$ independence queries and returns one of two useful structures: (i) b bases, each containing at least $m/2$ *private* elements, where the private subsets are pairwise disjoint; or (ii) a contraction minor that retains more than half of the rank and has at most $2b$ elements per unit of rank. Here an element is private to a base if it belongs to that base's designated private subset and to none of the other designated subsets. The bases themselves need not be disjoint.

The two outcomes correspond exactly to the two kinds of progress we need. In the first case, processing the bases as a batch can certify low elements from disjoint private subsets, ensuring that the same filtering progress is not counted repeatedly. In the second case, the same rank-reduction problem can be continued inside a substantially smaller minor while retaining a constant fraction of the original rank. Details are given in Appendix G.4 and Appendix G.5.

Turning private elements into distinct progress. Suppose **PackOrMinor** returns the first outcome. We process the bases as a batch. Whenever a basis contains sufficiently many high elements, one is chosen uniformly and added to S. Since this changes the residual matroid, each unprocessed basis is repaired by deleting a single element using the published independence-search primitive. If instead a basis contains few high elements, its low elements are recorded but their deletion is postponed until the end of the batch, so that filtering one basis does not invalidate the remaining bases before they are processed.

To quantify the progress, set $h_0 = \lceil 4\Gamma/\theta \rceil$. The density test guarantees that an acceptance occurs only when there are at least h_0 high candidates, while a filtering outcome leaves fewer than $2h_0$ high elements in the basis. Moreover, there can be at most $\Gamma/\theta \le h_0/4$ selections during an entire threshold pass, because each accepted element contributes marginal value at least θ. Hence only a small number of positions in each private subset can be lost to repairs before that basis is filtered. The density-test implementation and the corresponding value accounting are given in Appendix G.9.

3.2 Recursive rank reduction in smaller minors

The second outcome of **PackOrMinor** is a contraction minor that retains more than half of the current rank while containing far fewer elements per unit of rank. Rather than continuing to search in the full residual matroid, we recursively perform the same high-rank reduction task inside this smaller minor. Any elements selected by the recursive call are added to the same global prefix S.

When the call returns, the parent contracts these new selections and refreshes the set of high elements. Elements certified low in the child remain low after this update by diminishing returns. Auxiliary contractions used only to define the minor affect feasibility but are not added to S and do not enter the objective context.

The recursive call must guarantee slightly more rank reduction than its parent requires. We therefore organize the procedure into levels: a level-ℓ call aims to reduce the current high rank by a $2^{-\ell}$ fraction, while its child is called at level $\ell - 1$ and therefore targets twice that fractional reduction. Because the child retains more than half of the parent's current rank, this stronger reduction is sufficient to achieve the parent's target after the result is lifted back to the original residual matroid. Thus each parent needs at most one recursive child call, preventing uncontrolled branching.

The recursion also improves the query complexity. Let d denote the current rank and let q denote, up to constant factors, the number of active elements per unit of rank. A direct rebuild-and-filter procedure costs roughly $\widetilde{O}(q^2 d)$. If **PackOrMinor** returns b bases with disjoint private subsets, batching them reduces the repeated-scan contribution to roughly $\widetilde{O}((q^2/b)d)$. If instead it returns a smaller minor, that child has only $O(b)$ elements per unit of rank. If the previous recursion level has cost roughly $\widetilde{O}(q^p d)$, then processing the child costs roughly $\widetilde{O}(b^p d)$. Balancing the two branches, q^2/b and b^p, improves the exponent from p to $2p/(p+1)$.

Starting from the quadratic base case, the resulting exponents are $2, 4/3, 8/7, 16/15, \ldots$, and hence approach one rapidly. After $O(\log \log n)$ recursion levels, the remaining overhead is only polylogarithmic, yielding nearly linear query complexity. The formal recurrence, stopping conditions, and suppressed logarithmic factors are proved in Appendix G.6.

3.3 Completing the preprocessing guarantee

The outer preprocessing procedure repeats the rank-reduction task from the previous subsections across a decreasing sequence of marginal-value thresholds. At each threshold, it continues until the rank of the remaining high elements is sufficiently small. When the threshold is lowered, the algorithm restores the full residual ground set of M/S: an element that was unimportant at a higher threshold may still matter at a lower one, so filtering is temporary rather than permanent.

These rank bounds translate into control of the residual singleton marginals. Ordering the marginal values of any feasible residual set from largest to smallest, the threshold guarantees imply that the ith largest marginal decays proportionally to $1/i$ up to polylogarithmic factors. Summing this harmonic sequence yields the desired range bound in (3.1), with $\rho = O(\log^2(2n)\log(2r))$ for fixed threshold-spacing accuracy. The resulting prefix S is found using nearly linear value and independence queries. The preprocessing analysis also shows that the value gained by S compensates for the value it may block in a comparison optimum, which is what allows the prefix to be retained in the final solution rather than treated as a pure sampling device.

The next stage runs a range-sensitive continuous-greedy algorithm on the residual problem. If $a_e = f(e \mid S)$ bounds the range of a marginal sample for element e, we assign that element an additive estimation allowance proportional to a_e/ρ. Because the total of these singleton ranges along any feasible residual direction is at most $\rho\Gamma$, the additive errors across a direction sum to only $O(\nu\Gamma)$, while each individual marginal can still be estimated with a number of samples proportional to ρ. Thus, for $\nu = \Theta(\varepsilon)$, the residual fractional phase remains nearly linear.

This fractional phase produces a convex combination of $O(1/\varepsilon)$ residual bases. Adding the prefix S to each of them gives an explicit convex combination of bases of the original matroid, exactly in the form required by the rounding procedure of Section 2. Combining the preprocessing, residual fractional optimization, and rounding therefore yields the nearly linear independence-oracle algorithm. The full sampling guarantee and end-to-end composition are given in Appendix H.6.

4 Nearly Linear Optimization over Integral Polymatroids

We now turn to monotone DR-submodular maximization over an integral polymatroid. Unlike a matroid, a polymatroid may allow many copies of the same coordinate, and its numerical rank r

can be exponentially larger than the number of coordinate types n. Consequently, even logarithmic dependence on r can destroy near-linearity in n: for example, $\log^3 r = \Theta(n^3)$ when $r = 2^{\Theta(n)}$.

Representing a solution by its n coordinate counts avoids explicitly expanding all r units, but this alone is not enough. Three sources of rank dependence remain. First, marginal-value thresholds may have to span a range determined by very large coordinate capacities. Second, searching for the largest feasible count of a coordinate by ordinary binary search costs $\Theta(\log r)$ feasibility queries. Third, comparing a sampled direction with an optimum containing up to r units can introduce a $\log r$ factor through a union bound over those units. Our algorithm removes these three dependencies separately. Our main result is as follows.

Theorem 4.1 (Nearly linear optimization over integral polymatroids)**.** *Let $f : \mathbb{Z}_+^n \to \mathbb{R}_+$ be a normalized monotone DR-submodular function, and let P be an integral polymatroid of numerical rank r, accessed through an integer-feasibility oracle. For every $0 < \varepsilon \le 1/2$, there is an algorithm that always returns an integer feasible solution $X \in P$ satisfying $\mathbb{E}[f(X)] \ge (1 - 1/e - \varepsilon)\mathrm{OPT}$.*

The algorithm uses $O(n\varepsilon^{-9}\Xi_\varepsilon^3)$ value queries and $O(n\varepsilon^{-11}\Xi_\varepsilon^3)$ expected integer-feasibility queries, where $\Xi_\varepsilon = \log(2n/\varepsilon) + \log\log(r+3)$. In particular, the oracle complexity is nearly linear in the number of coordinate types even when the numerical rank is exponential in n.

At a high level, we first separate coordinate types by capacity scale and discard a small random fraction of scales. This produces an approximately feasible product relaxation in which different capacity groups can be processed independently. Within each group, we place a small integer baseline that reduces the range of all subsequent marginals. We then optimize the residual objective using large count increments rather than unit steps. Poisson smoothing and a small number of "colors" allow these large increments to capture nearly the full residual optimum gap. Geometrically spaced count searches reduce the number of feasibility tests to doubly logarithmic dependence on r, while aggregate concentration avoids paying separately for every unit of the comparison optimum. Finally, we shrink the fractional solution into the original polymatroid and round at most one remaining fractional unit per coordinate. Instead of the exact convex decomposition used by Soma and Yoshida (2018, Section 4.2.2), whose stated bound includes an $O(n^8)$ term, we construct an approximate cover by local matroid bases and apply our nearly linear rounder from Section 2. The local rounding stage uses $O(n\varepsilon^{-11}\log^3(2n))$ expected integer-feasibility queries and no value queries, hence $\widetilde{O}(n)$ oracle complexity for fixed ε.

Let $h = \lceil 16/\varepsilon \rceil$. The complete algorithm follows this structure:

1. group coordinate types by approximate capacity scale and form a product relaxation Q over the retained groups;
2. construct an integer baseline a and optimize the residual objective $g(x) = f(a + x) - f(a)$ over the residual region $Q_a = \{x \ge 0 : a + x \in Q\}$;
3. perform h^2 fractional updates, each obtained by building h color vectors whose sum is a feasible residual direction;
4. set $y = (1 - 1/h)(a + z)$, which lies in the original polymatroid; and
5. round y to an integer feasible solution using the local rounding procedure described in Section 4.2.

The full implementation and query accounting are given in Appendix I.2 and Appendix I.10 respectively.

4.1 Nearly linear fractional optimization at exponential rank

The main challenge is to construct a high-value fractional solution without paying polynomial or even logarithmic factors in the numerical rank at several different stages. We address these sources of rank dependence separately but within one fractional optimization procedure.

We first group coordinate types by approximate capacity scale, separated by small random gaps. Dropping the cross-group constraints produces a product relaxation that is only slightly larger than the original polymatroid, so a single final shrink restores feasibility. Within each retained group, we place a small integer baseline. By diminishing returns, this baseline bounds the remaining singleton marginals and makes the relevant threshold range independent of the absolute capacities. Together,

scale separation and the baseline eliminate the rank dependence that would otherwise arise from scanning marginal-value thresholds.

We then optimize the residual objective using large count increments. Each fractional direction is split into a small number of *colors*, which share one feasibility constraint but evaluate marginal gains in different objective contexts. This prevents diminishing returns within one bundle from making the greedy comparison too weak and allows each fractional step to capture nearly the full remaining optimum gap. Poisson smoothing provides the corresponding finite-step analysis, so the number of outer steps depends only on the desired accuracy rather than on the number of units represented by the direction.

Two additional devices keep the construction of each direction inexpensive. First, rather than searching all possible integer counts, we search a geometrically spaced grid. This reduces the number of feasibility tests for a coordinate to doubly logarithmic dependence on the numerical rank. Analytical shadow counts compensate for the gaps between consecutive grid points and restore the rejection certificates needed in the comparison proof. Second, we control sampling error in aggregate rather than requiring every unit of an unknown optimum to be estimated accurately. The resulting concentration bound depends on the number of actual sampled tests rather than the potentially exponential number of optimum units.

Combining these ingredients yields a fractional point with expected value within $O(\epsilon)$ of the continuous-greedy guarantee using nearly linear oracle complexity in the number of coordinate types and only doubly logarithmic dependence on the numerical rank. The complete scale-separation argument, Poisson and color analysis, geometric-search certificates, and aggregate concentration proof are given in Appendix I.

4.2 Returning to the original polymatroid and rounding

The fractional construction takes place in the product relaxation Q. By construction, the averaged residual point satisfies $a+z \in Q$ on every run. We therefore shrink once and set $y = (1-1/h)(a+z)$. Lemma I.3 guarantees that $y \in P$ unconditionally. Thus feasibility of the fractional point does not depend on the sampling event used in the value analysis.

It remains to convert y into an integer feasible solution. The integer part $\lfloor y \rfloor$ can be fixed immediately. Each coordinate then has at most one remaining fractional unit, so there are at most n rounding decisions regardless of the numerical rank. These round-up choices form a local matroid, as in the reduction of Soma and Yoshida (2018).

Previous approaches obtain an exact convex decomposition of the fractional part before applying matroid rounding. We avoid this expensive step. Let $v = y - \lfloor y \rfloor$. It is enough to construct local bases whose average covers at least a $(1-\alpha)$ fraction of v coordinatewise, since any additional coverage can only improve a monotone objective. A multiplicative-weights procedure repeatedly chooses a maximum-weight local base and produces such a covering using only a number of bases polynomial in the denominator and the accuracy. Because the fractional construction above gives denominator $D = h^3$, this list size depends only on the accuracy and logarithmic factors, not on the numerical rank.

We then pass this short base list to the independence-oracle rounder from Section 2. The resulting integer vector is feasible in the original polymatroid on every run and loses only another $O(1/h)$ fraction of the local fractional value. The complete local-rounding guarantee is proved in Appendix I.9.

Combining random scale retention, the baseline, the fractional progress argument, the final shrink, and local rounding gives an expected $(1-1/e-\varepsilon)$ approximation. The algorithm uses $O(n\varepsilon^{-9}\Xi_\varepsilon^3)$ value queries and $O(n\varepsilon^{-11}\Xi_\varepsilon^3)$ expected integer-feasibility queries, where $\Xi_\varepsilon = \log(2n/\varepsilon)+\log\log(r+3)$. Hence the oracle complexity remains nearly linear in the number of coordinate types even when $r = 2^{\Theta(n)}$. All returned solutions are feasible on every run; the full query accounting and approximation proof appear in Appendix I.10.

REPRODUCIBILITY STATEMENT

The appendices specify the oracle models and all assumptions, give complete pseudocode and parameter choices, identify the exact source locations of published subroutines, and provide proofs of the approximation and query bounds. The supplementary guide below maps the main-text mechanisms to their formal statements. Numerical examples are illustrations of the analysis, not empirical runtime measurements. The results concern oracle complexity; no claim about an implemented algorithm's wall-clock performance is made.

AI USE STATEMENT

Generative AI tools assisted in preparing earlier versions of this work and this manuscript, including development of mathematical arguments, proof writing and explanation, literature synthesis, organization of the technical presentation, and LaTeX preparation. AI tools also assisted with checking numerical examples, cross-references, and document layout. These computational and editorial checks do not replace mathematical verification of the claims.

REFERENCES

Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. "Matroids, secretary problems, and online mechanisms." In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 434–443, 2007.

Moshe Babaioff, Jason Hartline, and Robert Kleinberg. "Selling banner ads: Online algorithms with buyback." In *Fourth Workshop on Ad Auctions*, 2008.

Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H. Chi, and Jennifer Gillenwater. "Practical diversified recommendations on YouTube with determinantal point processes." In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2165–2173, 2018.

Zhou Xu and Brian Rodrigues. "A 3/2-approximation algorithm for the multiple TSP with a fixed number of depots." *INFORMS Journal on Computing*, 27(4):636–645, 2015.

M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. "An analysis of approximations for maximizing submodular set functions–II." *Mathematical Programming Studies*, 8:73–87, 1978.

Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. "Maximizing a monotone submodular function subject to a matroid constraint." *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

Ashwinkumar Badanidiyuru and Jan Vondrák. "Fast algorithms for maximizing submodular functions." In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1497–1514, 2014.

Monika Henzinger, Paul Liu, Jan Vondrák, and Da Wei Zheng. "Faster submodular maximization for several classes of matroids." *arXiv preprint arXiv:2305.00122*, 2023.

Uriel Feige. "A threshold of ln n for approximating set cover." *Journal of the ACM (JACM)*, vol. 45, no. 4, pp. 634–652, 1998.

Rohan Ghuge, Anupam Gupta, and Viswanath Nagarajan. "The power of adaptivity for stochastic submodular cover." In *International Conference on Machine Learning*, pp. 3702–3712, PMLR, 2021.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. "Glister: Generalization based data subset selection for efficient and robust learning." In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 8110–8118, 2021.

Baharan Mirzasoleiman, Stefanie Jegelka, and Andreas Krause. "Streaming non-monotone submodular maximization: Personalized video summarization on the fly." In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Yixin Chen, Ankur Nath, Chunli Peng, and Alan Kuhnle. "Discretely beyond $1/e$: Guided Combinatorial Algortihms for Submodular Maximization." In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. "Non-monotone submodular maximization under matroid and knapsack constraints." In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 323–332, 2009.

Benjamin Qi. "On maximizing sums of non-monotone submodular and linear functions." *Algorithmica*, vol. 86, no. 4, pp. 1080–1134, 2024.

Shayan Oveis Gharan and Jan Vondrák. "Submodular maximization by simulated annealing." In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pp. 1098–1116, SIAM, 2011.

Niv Buchbinder and Moran Feldman. "Constrained submodular maximization via new bounds for dr-submodular functions." In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pp. 1820–1831, 2024.

Niv Buchbinder and Moran Feldman. "Constrained submodular maximization via a nonsymmetric technique." *Mathematics of Operations Research*, vol. 44, no. 3, pp. 988–1005, 2019.

Moran Feldman, Joseph Naor, and Roy Schwartz. "A unified continuous greedy algorithm for submodular maximization." In *2011 IEEE 52nd annual symposium on foundations of computer science*, pp. 570–579, IEEE, 2011.

Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. "Streaming submodular maximization: Massive data summarization on the fly." In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 671–680, ACM, 2014.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. "An analysis of approximations for maximizing submodular set functions—I." *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

Eric Balkanski, Adam Breuer, and Yaron Singer. "Non-monotone submodular maximization in exponentially fewer iterations." *Advances in Neural Information Processing Systems*, vol. 31, 2018.

Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. "Learning mixtures of submodular functions for image collection summarization." *Advances in neural information processing systems*, vol. 27, 2014.

David Kempe, Jon Kleinberg, and Éva Tardos. "Maximizing the spread of influence through a social network." In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, ACM, 2003.

MohammadHossein Bateni, Hossein Esfandiari, and Vahab Mirrokni. "Almost optimal streaming algorithms for coverage problems." In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 13–23, 2017.

Alessandro Epasto, Vahab Mirrokni, and Morteza Zadimoghaddam. "Bicriteria distributed submodular maximization in a few rounds." In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 25–33, 2017.

Rishabh K Iyer and Jeff A Bilmes. "Submodular optimization with submodular cover and submodular knapsack constraints." *Advances in neural information processing systems*, vol. 26, 2013.

Ashkan Norouzi-Fard, Abbas Bazzi, Ilija Bogunovic, Marwa El Halabi, Ya-Ping Hsieh, and Volkan Cevher. "An efficient streaming algorithm for the submodular cover problem." *Advances in Neural Information Processing Systems*, vol. 29, 2016.

Chris Harshaw, Moran Feldman, Justin Ward, and Amin Karbasi. "Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications." In *International Conference on Machine Learning*, pp. 2634–2643, PMLR, 2019.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. "Lazier than lazy greedy." In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

Laurence A Wolsey. "An analysis of the greedy algorithm for the submodular set covering problem." *Combinatorica*, vol. 2, no. 4, pp. 385–393, 1982.

Baharan Mirzasoleiman, Amin Karbasi, Ashwinkumar Badanidiyuru, and Andreas Krause. "Distributed submodular cover: Succinctly summarizing massive data." *Advances in Neural Information Processing Systems*, vol. 28, 2015.

Anupam Gupta and Roie Levin. "The online submodular cover problem." In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1525–1537, SIAM, 2020.

Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta. "Robust Submodular Observation Selection.." *Journal of Machine Learning Research*, vol. 9, no. 12, 2008.

Naoto Ohsaka and Tatsuya Matsuoka. "Approximation algorithm for submodular maximization under submodular cover." In *Uncertainty in Artificial Intelligence*, pp. 792–801, PMLR, 2021.

Tasuku Soma and Yuichi Yoshida. "A generalization of submodular cover via the diminishing return property on the integer lattice." *Advances in neural information processing systems*, vol. 28, 2015.

Rishabh Iyer and Jeffrey Bilmes. "Near optimal algorithms for hard submodular programs with discounted cooperative costs." In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 276–285, PMLR, 2019.

Rishabh Iyer. "A Unified Framework of Constrained Robust Submodular Optimization with Applications." *arXiv preprint arXiv:1906.06393*, 2019.

Ashwinkumar Badanidiyuru and Jan Vondrák. "Fast algorithms for maximizing submodular functions." In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1497–1514, SIAM, 2014.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. "Submodular maximization with cardinality constraints." In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1433–1452, SIAM, 2014.

Yisong Yue and Carlos Guestrin. "Linear submodular bandits and their application to diversified retrieval." *Advances in Neural Information Processing Systems*, vol. 24, 2011.

Niv Buchbinder, Moran Feldman, Joseph Seffi, and Roy Schwartz. "A tight linear time (1/2)-approximation for unconstrained submodular maximization." *SIAM Journal on Computing*, vol. 44, no. 5, pp. 1384–1402, 2015.

Morteza Monemizadeh. "Dynamic submodular maximization." *Advances in Neural Information Processing Systems*, vol. 33, pp. 9806–9817, 2020.

Erik M Lindgren, Shanshan Wu, and Alexandros G Dimakis. "Sparse and greedy: Sparsifying submodular facility location problems." In *NIPS Workshop on Optimization for Machine Learning*, 2015.

Zheng Wen, Branislav Kveton, Michal Valko, and Sharan Vaswani. "Online influence maximization under independent cascade model with semi-bandit feedback." *Advances in neural information processing systems*, vol. 30, 2017.

Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. "Maximizing a monotone submodular function subject to a matroid constraint." *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

Wenjing Chen, Shuo Xing, and Victoria G Crawford. "A Threshold Greedy Algorithm for Noisy Submodular Maximization." *arXiv preprint arXiv:2312.00155*, 2023.

Manish Prajapat, Mojmír Mutnỳ, Melanie N Zeilinger, and Andreas Krause. "Submodular reinforcement learning." *arXiv preprint arXiv:2307.13372*, 2023.

Niv Buchbinder and Moran Feldman. "Deterministic algorithm and faster algorithm for submodular maximization subject to a matroid constraint." In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 700–712, IEEE, 2024.

Satoru Iwata, Shin-ichi Tanigawa, and Yuichi Yoshida. "Improved approximation algorithms for k-submodular function maximization." In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pp. 404–413, SIAM, 2016.

Marwa El Halabi, Slobodan Mitrović, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub M Tarnawski. "Fairness in streaming submodular maximization: Algorithms and hardness." *Advances in Neural Information Processing Systems*, vol. 33, pp. 13609–13622, 2020.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. "Gptq: Accurate post-training quantization for generative pre-trained transformers." *arXiv preprint arXiv:2210.17323*, 2022.

Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. "An analysis of approximations for maximizing submodular set functions—II." Springer, 1978.

Chandra Chekuri and Kent Quanrud. "Submodular function maximization in parallel via the multilinear relaxation." In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 303–322, SIAM, 2019.

Marwa El Halabi, Federico Fusco, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub Tarnawski. "Fairness in streaming submodular maximization over a matroid constraint." In *International Conference on Machine Learning*, pp. 9150–9171, PMLR, 2023.

Manish Prajapat, Mojmír Mutnỳ, Melanie N Zeilinger, and Andreas Krause. "Submodular reinforcement learning." *arXiv preprint arXiv:2307.13372*, 2023.

Gaurush Hiranandani, Harvineet Singh, Prakhar Gupta, Iftikhar Ahamath Burhanuddin, Zheng Wen, and Branislav Kveton. "Cascading linear submodular bandits: Accounting for position bias and diversity in online learning to rank." In *Uncertainty in Artificial Intelligence*, pp. 722–732, PMLR, 2020.

Lifeng Zhou and Pratap Tokekar. "Risk-aware submodular optimization for multirobot coordination." *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3064–3084, 2022.

Baijun Ji, Xiangyu Duan, Zhenyu Qiu, Tong Zhang, Junhui Li, Hao Yang, and Min Zhang. "Submodular-based In-context Example Selection for LLMs-based Machine Translation." In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 15398–15409, 2024.

Lilly Kumari, Shengjie Wang, Arnav Das, Tianyi Zhou, and Jeff Bilmes. "An End-to-End Submodular Framework for Data-Efficient In-Context Learning." In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3293–3308, 2024.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. "Submodular subset selection for large-scale speech training data." In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3311–3315, IEEE, 2014.

Marwa El Halabi, Suraj Srinivas, and Simon Lacoste-Julien. "Data-efficient structured pruning via submodular optimization." *Advances in Neural Information Processing Systems*, vol. 35, pp. 36613–36626, 2022.

Satoki Tsuji, Fuyuka Yamada, Hiroshi Kawaguchi, Atsuki Inoue, and Yasufumi Sakai. "Greedy search algorithm for partial quantization of convolutional neural networks inspired by submodular optimization." *Neural Computing and Applications*, pp. 1–11, 2022.

Jiaxing Shang, Shangbo Zhou, Xin Li, Lianchen Liu, and Hongchun Wu. "CoFIM: A community-based framework for influence maximization on large-scale networks." *Knowledge-Based Systems*, vol. 117, pp. 88–100, 2017.

Wei Chen, Chi Wang, and Yajun Wang. "Scalable influence maximization for prevalent viral marketing in large-scale social networks." In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1029–1038, 2010.

Naoto Ohsaka and Yuichi Yoshida. "Monotone k-submodular function maximization with size constraints." *Advances in Neural Information Processing Systems*, vol. 28, 2015.

Lan Nguyen and My T Thai. "Streaming k-submodular maximization under noise subject to size constraint." In *International conference on machine learning*, pp. 7338–7347, PMLR, 2020.

Wenjing Chen and Victoria Crawford. "Bicriteria approximation algorithms for the submodular cover problem." *Advances in Neural Information Processing Systems*, vol. 36, 2024.

Victoria G Crawford. "An efficient evolutionary algorithm for minimum cost submodular cover." In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 1227–1233, 2019.

Pinar Duygulu, Kobus Barnard, Joao FG de Freitas, and David A Forsyth. "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary." In *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part IV 7*, pp. 97–112, Springer, 2002.

Niv Buchbinder and Moran Feldman. "Deterministic algorithms for submodular maximization problems." *ACM Transactions on Algorithms (TALG)*, vol. 14, no. 3, pp. 1–20, 2018.

Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. "Distributed submodular maximization." *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 8330–8373, 2016.

Kiarash Banihashem, Samira Goudarzi, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Morteza Monemizadeh. "A dynamic algorithm for weighted submodular cover problem." *arXiv preprint arXiv:2407.10003*, 2024.

Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. "Maximizing nonmonotone submodular functions under matroid or knapsack constraints." *SIAM Journal on Discrete Mathematics*, vol. 23, no. 4, pp. 2053–2078, 2010.

Alina Ene, Huy L Nguyen, and Adrian Vladu. "Submodular maximization with matroid and packing constraints in parallel." In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, pp. 90–101, 2019.

Moran Feldman, Amin Karbasi, and Ehsan Kazemi. "Do less, get more: Streaming submodular maximization with subsampling." *Advances in Neural Information Processing Systems*, vol. 31, 2018.

Kai Han, Shuang Cui, Benwei Wu, and others. "Deterministic approximation for submodular maximization over a matroid in nearly linear time." *Advances in Neural Information Processing Systems*, vol. 33, pp. 430–441, 2020.

Jure Leskovec and Rok Sosič. "Snap: A general-purpose network analysis and graph-mining library." *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 1, pp. 1–20, 2016.

Victoria Crawford, Alan Kuhnle, and My Thai. "Submodular cost submodular cover with an approximate oracle." In *International Conference on Machine Learning*, pp. 1426–1435, PMLR, 2019.

Laurence A. Wolsey. "An analysis of the greedy algorithm for the submodular set covering problem." *Comb.*, vol. 2, no. 4, pp. 385–393, 1982.

Benedek Rozemberczki and Rik Sarkar. "Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings." *arXiv preprint arXiv:2101.03091*, 2021.

Victoria Crawford. "Scalable bicriteria algorithms for non-monotone submodular cover." In *International Conference on Artificial Intelligence and Statistics*, pp. 9517–9537, PMLR, 2023.

Wenjing Chen, Shuo Xing, Samson Zhou, and Victoria G Crawford. "Fair Submodular Cover." In *The Thirteenth International Conference on Learning Representations*, 2025.

Rishabh Iyer and Jeffrey Bilmes. "Submodular point processes with applications to machine learning." In *Artificial Intelligence and Statistics*, pp. 388–397, PMLR, 2015.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

Ishika Agarwal, Krishnateja Killamsetty, Lucian Popa, and Marina Danilevsky. "DELIFT: Data Efficient Language model Instruction Fine-Tuning." In *The Thirteenth International Conference on Learning Representations*, 2024.

Wenjing Chen and Victoria G Crawford. "Linear Submodular Maximization with Bandit Feedback." In *International Conference on Artificial Intelligence and Statistics*, pp. 4249–4257, PMLR, 2025.

Zirui Xu and Vasileios Tzoumas. "Performance-aware self-configurable multi-agent networks: A distributed submodular approach for simultaneous coordination and network design." In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pp. 5393–5400, IEEE, 2024.

Ruoyu Chen, Hua Zhang, Siyuan Liang, Jingzhi Li, and Xiaochun Cao. "Less is More: Fewer Interpretable Region via Submodular Subset Selection." In *The Twelfth International Conference on Learning Representations*, 2024.

Georgios Amanatidis, Federico Fusco, Philip Lazos, Stefano Leonardi, and Rebecca Reiffenhäuser. "Fast adaptive non-monotone submodular maximization subject to a knapsack constraint." *Advances in neural information processing systems*, vol. 33, pp. 16903–16915, 2020.

Shuang Cui, Kai Han, Shaojie Tang, Feng Li, and Jun Luo. "Fairness in streaming submodular maximization subject to a knapsack constraint." In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 514–525, 2024.

Lijun Li, Chenyang Xu, Liuyi Yang, and Ruilong Zhang. "Fair Submodular Maximization over a Knapsack Constraint." In *34th International Joint Conference on Artificial Intelligence (2025)*, pp. 3934–3942, International Joint Conferences on Artificial Intelligence, 2025.

Shuang Cui, Kai Han, and Jing Tang. "Linear-Time Algorithms for Representative Subset Selection From Data Streams." In *Proceedings of the ACM on Web Conference 2025*, pp. 4710–4721, 2025.

S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing*, 8:121–164, 2012. https://doi.org/10.4086/toc.2012.v008a006.

A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *SODA 2014*, pp. 1497–1514, 2014. https://doi.org/10.1137/1.9781611973402.110.

E. Balkanski, A. Rubinstein, and Y. Singer. An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model. In *STOC 2019*, pp. 66–77, 2019. https://doi.org/10.1145/3313276.3316304. Full version: https://arxiv.org/abs/1811.03093v1; Section 4 treats rank and independence access separately.

S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013. Chapter 2, especially the bounded-sum Chernoff and Bernstein inequalities. https://doi.org/10.1093/acprof:oso/9780199535255.001.0001.

N. Buchbinder and M. Feldman. Deterministic algorithm and faster algorithm for submodular maximization subject to a matroid constraint. *SIAM Journal on Computing*, published online 2025; conference version in *FOCS 2024*. https://doi.org/10.1137/24M1698122. Source numbering: https://arxiv.org/abs/2408.03583v3.

N. Buchbinder and M. Feldman. Extending the extension: Deterministic algorithm for non-monotone submodular maximization. In *STOC 2025*, pp. 1130–1141, 2025. https://doi.org/10.1145/3717823.3718120. Source numbering: https://arxiv.org/abs/2409.14325v1.

N. Buchbinder, M. Feldman, and R. Schwartz. Comparing apples and oranges: Query trade-off in submodular maximization. *Mathematics of Operations Research*, 42(2):308–329, 2017. https://doi.org/10.1287/moor.2016.0809. Source numbering: https://arxiv.org/abs/1410.0773.

G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011. https://doi.org/10.1137/080733991.

C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS 2010*, pp. 575–584, 2010. https://doi.org/10.1109/FOCS.2010.60.

J. Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Structures and Their Applications*, pp. 69–87. Gordon and Breach, 1970.

J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1:127–136, 1971. https://doi.org/10.1007/BF01584082.

A. Ene and H. L. Nguyen. A reduction for optimizing lattice submodular functions with diminishing returns. Preprint, 2016, revised 2018. https://arxiv.org/abs/1606.08362.

A. Ene and H. L. Nguyen. Towards nearly-linear time algorithms for submodular maximization with a matroid constraint. In *ICALP 2019*, LIPIcs 132, Article 54, 2019. https://doi.org/10.4230/LIPIcs.ICALP.2019.54. Source numbering: https://arxiv.org/abs/1811.07464v1, Lemmas 7 and 12–14; see also Kobayashi and Terao (2024, Lemma 6.1).

A. Ganz Rozenman, A. Kulik, R. Schwartz, and M. Singh. A Poisson process for submodular maximization. Preprint, May 2026. https://arxiv.org/abs/2605.03071v1.

M. Henzinger, P. Liu, J. Vondrák, and D. W. Zheng. Faster submodular maximization for several classes of matroids. In *ICALP 2023*, LIPIcs 261, Article 74, 2023. https://doi.org/10.4230/LIPIcs.ICALP.2023.74. https://arxiv.org/abs/2305.00122.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. https://doi.org/10.1080/01621459.1963.10500830.

R. M. Karp, E. Upfal, and A. Wigderson. The complexity of parallel search. *Journal of Computer and System Sciences*, 36(2):225–253, 1988. Author's journal-version archive: https://www.math.ias.edu/avi/node/765.

Y. Kobayashi and T. Terao. Subquadratic submodular maximization with a general matroid constraint. In *ICALP 2024*, LIPIcs 297, Article 100, 2024. https://doi.org/10.4230/LIPIcs.ICALP.2024.100. Source numbering: https://arxiv.org/abs/2405.00359v1.

A. Kulik, T. Oliveira, R. Schwartz, and M. Singh. If it is good then drop it – a spiteful Poisson process for submodular maximization. Preprint, August 2026. https://arxiv.org/abs/2608.05062v1.

S. Nematollahi, A. Vladu, and J. Zhao. Fixed-parameter tractable submodular maximization over a matroid. In *ITCS 2026*, LIPIcs 362, Article 105, 2026. https://doi.org/10.4230/LIPIcs.ITCS.2026.105.

G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3(3):177–188, 1978. https://doi.org/10.1287/moor.3.3.177.

J. Oxley. *Matroid Theory*, second edition. Oxford University Press, 2011.

B. Peng and A. Rubinstein. A near linear query lower bound for submodular maximization. In *ICML 2025*, PMLR 267, pp. 48852–48871, 2025. https://proceedings.mlr.press/v267/peng25d.html. Theorems 3.7 and 3.9 give the search and optimum-value lower bounds.

K. Quanrud. Faster exact and approximation algorithms for packing and covering matroids via push–relabel. In *SODA 2024*, pp. 2305–2336, 2024. Source numbering: https://arxiv.org/abs/2303.01478v1.

T. Soma and Y. Yoshida. Maximizing monotone submodular functions over the integer lattice. *Mathematical Programming*, 172:539–563, 2018; conference version in *IPCO 2016*. https://doi.org/10.1007/s10107-018-1324-y. Source numbering and complexity comparison: https://arxiv.org/abs/1503.01218v2, Section 4.

J. Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM Journal on Computing*, 42(1):265–304, 2013. https://doi.org/10.1137/110832318. Full version: https://arxiv.org/abs/1110.4860; see Theorem 1.8 and Section 2 for the randomized oracle barrier.

SUPPLEMENTARY MATERIAL: COMPLETE ALGORITHMS AND PROOFS

The main text follows the algorithmic dependencies. These appendices give the detailed statements, implementation choices, examples, and proofs. Published tools are identified by their exact source locations; all new claims used in the main text are proved here.

Appendix	Contents
A	Definitions, oracle models, and elementary matroid facts
B	Detailed prior-work comparison and lower-bound consequence
C	Full result statements and the previous framework's costs
D	Published rank-oracle fractional phase and its guarantees
E	Fixed-length walk rounding and endpoint analysis
F	Geometric rounding and the global removal budget
G	Private subsets, small minors, and multilevel preprocessing
H	Residual sampled continuous greedy and composition
I	One direct polymatroid algorithm: capacity scales, baseline, Poisson colors, geometric search, aggregate errors, and local rounding

The private-subset alternative is Lemma G.3; its recursive cost is Lemma G.10. The polymatroid construction is Algorithm I.1, presented before its supporting arguments. Lemmas I.3–I.5 prove scale separation and baseline preservation. Lemma I.13 supplies Poisson progress. Lemmas I.16, I.17, and I.18 justify geometric search and aggregate errors, followed by the direction gain in Lemma I.19. The local rounder is Theorem I.25; the end-to-end proof and original-oracle account are in Appendix I.10. Proposition I.26 gives the exponential-rank lower bound.

The appendices develop this final algorithm directly. The local rounding proof applies to arbitrary feasible rational points and is independent of the fractional construction; its earlier use for baseline preservation is an existence argument, not an extra algorithmic call. No unpublished companion manuscript is required to check the guarantees.

A CONCEPTS, EXAMPLES, AND THE ORACLE MODEL

We seek a high-value feasible solution when additional choices have diminishing returns. A feasible solution is a set in the matroid problem and a vector of nonnegative integer multiplicities in the polymatroid problem. The distinction matters: the number of coordinate types can be small even when a feasible integer solution contains an enormous number of units. We begin with the definitions used throughout the paper.

A.1 SUBMODULAR SET FUNCTIONS AND DIMINISHING RETURNS

Definition A.1 (Set submodularity)**.** A function $f : 2^V \to \mathbb{R}$ is submodular if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \qquad (A, B \subseteq V).$$

Equivalently, for $A \subseteq B$ and $e \notin B$, $f(A+e) - f(A) \geq f(B+e) - f(B)$. It is monotone if $A \subseteq B$ implies $f(A) \leq f(B)$, and normalized if $f(\varnothing) = 0$. We write $f(e \mid S) = f(S + e) - f(S)$ and $f(T \mid S) = f(S \cup T) - f(S)$.

The equivalence follows by adding the elements of a set difference one at a time. The diminishing-returns inequality bounds each later marginal by its earlier counterpart; summing gives the two-set inequality. Conversely, apply the two-set inequality to $A + e$ and B.

Example A.2 (Overlapping coverage)**.** Let $C_a = \{1, 2\}$, $C_b = \{2, 3\}$, and $C_c = \{1, 3\}$, and set $f(S) = |\bigcup_{e \in S} C_e|$. Then $f(a \mid \varnothing) = 2$ but $f(a \mid \{b\}) = 1$. Each item can cover only features not already covered; this proves diminishing returns for arbitrary nonnegative weighted coverage. Singleton sums can nevertheless greatly overestimate feasible-set value. If r items all cover one feature, their singleton sum is r and the value of their union is one. This discrepancy is a central sampling bottleneck.

All set objectives in the paper are nonnegative and monotone. We may normalize by replacing f with $f - f(\varnothing)$. If an algorithm gives ratio $a \leq 1$ for the normalized function, then its original value is at least $a\mathrm{OPT} + (1 - a)f(\varnothing) \geq a\mathrm{OPT}$. Thus normalization does not weaken our guarantees.

A.2 MATROIDS, BASES, RANK, AND EXCHANGE

Definition A.3 (Matroid)**.** A matroid $M = (V, \mathcal{I})$ consists of a finite ground set V and a family of independent sets $\mathcal{I} \subseteq 2^V$ such that

(i) $\varnothing \in \mathcal{I}$;

(ii) $I \in \mathcal{I}$ and $J \subseteq I$ imply $J \in \mathcal{I}$;

(iii) if $I, J \in \mathcal{I}$ and $|I| < |J|$, then some $e \in J \setminus I$ satisfies $I + e \in \mathcal{I}$.

A base is an inclusion-maximal independent set. The rank of $A \subseteq V$ is $r_M(A) = \max\{|I| : I \subseteq A,\ I \in \mathcal{I}\}$. The rank of M is $r = r_M(V)$.

Augmentation implies that all bases have size r and that every independent set extends to a base. It also implies the basis-exchange property: for bases B_1, B_2 and $u \in B_1 \setminus B_2$, some $v \in B_2 \setminus B_1$ makes $B_1 - u + v$ a base. This is a one-sided exchange; a two-sided exchange also requires $B_2 - v + u$ to be a base.

Example A.4 (Three standard matroids)**.** In the uniform matroid $U_{r,n}$, all sets of size at most r are independent. In a partition matroid, V is partitioned into classes V_j and independence means $|I \cap V_j| \leq b_j$ for every class. In a graphic matroid, the elements are edges of a graph and independent sets are forests; bases are spanning forests. For a triangle with edges a, b, c, every two-edge set is a base, $r_M(\{a, b, c\}) = 2$, and the three-edge set is dependent. A linear matroid instead represents elements by vectors and calls a set independent when its vectors are linearly independent. These examples motivate the abstract axioms, but none of our algorithms assumes a graph, partition, or matrix representation.

A circuit is a minimal dependent set. If B is a base and $e \notin B$, $B + e$ contains a unique circuit, the fundamental circuit $C_B(e)$. For $b \in B$, $B - b + e$ is a base exactly when $b \in C_B(e) \setminus \{e\}$.

We use the standard circuit and closure properties of matroids; see (Oxley, 2011). The closure is $\mathrm{cl}(A) = \{e : r_M(A + e) = r_M(A)\}$.

Definition A.5 (Restriction and contraction)**.** The restriction $M|_X$ retains only elements of $X \subseteq V$. For an independent set S, the contraction M/S is the matroid on $V \setminus S$ in which I is independent exactly when $S \cup I$ is independent in M.

One original independence query simulates one query in a restriction or a contraction by a known independent set. In the recursion, some contractions simplify feasibility without committing those elements to the objective. The algorithms will keep the selected prefix separate from these auxiliary contractions.

Proposition A.6 (Maximum-weight base by matroid greedy)**.** *Given explicit real weights on N elements and an independence oracle,* **MaximumWeightBase** *returns a maximum-weight base using at most N independence queries. It orders elements by decreasing weight and accepts an element when independence is preserved. This is the classical matroid greedy theorem (Oxley, 2011); sorting uses no objective-value query.*

A.3 Rank oracles and independence oracles

An oracle specifies what information an algorithm can request about a constraint without requiring a particular graph, matrix, or other representation. Listing all independent sets would already be exponentially large for many simple matroids. Instead we supply one of two routines:

- An *independence oracle* receives an arbitrary subset $A \subseteq V$ and returns one bit: whether $A \in \mathcal{I}$. It says whether every element in the proposed set can be kept.
- A *rank oracle* receives the same kind of subset and returns $r_M(A)$, the maximum number of its elements that can be kept independently. It does not have to return a maximizing subset.

One rank query answers independence by testing $r_M(A) = |A|$. Conversely, scanning A and greedily retaining each element that preserves independence computes $r_M(A)$ with at most $|A|$ independence queries. Augmentation guarantees that the resulting maximal independent subset has maximum size. The two oracles are therefore polynomially interconvertible, but replacing every rank query by this scan need not preserve a nearly linear query bound.

Where these models come from. They belong to classical matroid algorithms, rather than originating in submodular maximization. Edmonds' greedy framework (Edmonds, 1971) uses the ability to test independence; Karp–Upfal–Wigderson (Karp et al., 1988) explicitly study the different powers of rank and independence access for parallel search. Within submodular maximization itself, Balkanski–Rubinstein–Singer (Balkanski et al., 2019, Section 4) distinguish their parallel-round complexities before Kobayashi–Terao (Kobayashi and Terao, 2024) compare the sequential query counts relevant here. These references establish the earlier use of the models; we do not attribute the first use of the term "rank oracle" to a recent submodular-maximization paper.

Example A.7 (Why the stronger oracle matters)**.** Let J be an independent set of size $r - 1$ and let T contain candidate entrants. The predicate $r_M(J \cup T) = r$ says that some candidate can extend J. It supports binary search even when $J \cup T$ is oversized. An independence query to that oversized set only says "dependent". For instance, in a rank-two linear matroid let $J = \{(1, 0)\}$. Two additional vectors parallel to $(1, 0)$ give rank one, whereas including $(0, 1)$ gives rank two. Both three-element sets are dependent. An independence oracle cannot distinguish them with that one query. With 1024 candidates, rank access can locate an extending element using an initial existence test and ten halving tests. This is a group-existence test, not a request to accept all 1024 candidates at once. Independence access requires a different search construction, as our rounding algorithm will illustrate.

A value oracle returns $f(A)$ for any subset, feasible or infeasible. Each query to either supplied oracle costs one unit in our complexity statements. Explicit set or vector construction and numerical bit complexity are separate costs. In particular, constructing or transmitting a dense query argument is not included in one unit of oracle cost. We use $\widetilde{O}$ only to suppress logarithmic factors, and retain polynomial versus exponential dependence on $1/\varepsilon$ explicitly.

A.4 POLYMATROIDS AND INTEGER-LATTICE OBJECTIVES

Definition A.8 (Integral polymatroid)**.** Let $\rho : 2^E \to \mathbb{Z}_+$ be normalized, monotone, and submodular. Its polymatroid is

$$P(\rho) = \{x \in \mathbb{R}_+^E : x(A) \le \rho(A) \text{ for every } A \subseteq E\}, \qquad x(A) = \sum_{e \in A} x_e.$$

Its rank is $r = \rho(E)$; the capacity of coordinate e is $u_e = \rho(\{e\})$. We maximize over $P(\rho) \cap \mathbb{Z}_+^E$ and write $n = |E|$ for the number of coordinate types, not the total number of units.

Edmonds' polymatroid theory (Edmonds, 1970) gives integrality and the greedy linear-optimization characterization. A matroid independence polytope is the special case in which all singleton ranks are at most one:

$$P(M) = \operatorname{conv}\{\mathbf{1}_I : I \in \mathcal{I}\} = \{x \ge 0 : x(A) \le r_M(A) \text{ for all } A \subseteq V\}.$$

Its base polytope additionally requires $x(V) = r$. For a general integral polymatroid, r need not be at most n.

Example A.9 (Two types and five available units)**.** Let $\rho(\{1\}) = 3$, $\rho(\{2\}) = 4$, and $\rho(\{1,2\}) = 5$. Then $P(\rho)$ is given by $x_1 \le 3$, $x_2 \le 4$, and $x_1 + x_2 \le 5$. The point $(3,2)$ is feasible and $(3,3)$ is not. Here $n = 2$ and $r = 5$; replacing 5 by a large numerical capacity makes the distinction between n and r essential. The integer-feasibility oracle answers whether a supplied count vector satisfies all inequalities, without revealing which inequality fails. It does not return $\rho(A)$.

Definition A.10 (Lattice diminishing returns)**.** A function $f : \mathbb{Z}_+^E \to \mathbb{R}_+$ is DR-submodular if

$$\Delta_e f(x) := f(x + \mathbf{1}_e) - f(x) \ge \Delta_e f(y) \quad \text{for all } x \le y \text{ and } e \in E.$$

We assume monotonicity and $f(0) = 0$. The condition includes diminishing returns from repeatedly increasing the same coordinate. It is stronger than the lattice inequality $f(x) + f(y) \ge f(x \vee y) + f(x \wedge y)$.

For example, $f(x) = \min\{3, 2x_1 + x_2\} + \min\{2, x_2\}$ is monotone DR. Each summand is a concave function of a nonnegative modular quantity. The lattice value oracle is defined on all of $\mathbb{Z}_+^E$, including infeasible points. This is needed for the sampled translations in the polymatroid algorithm. Constraints use only integer-feasibility access. Neither the rank r nor the singleton capacities need be supplied. The algorithm in Section I needs only approximate capacity scales and never evaluates the polymatroid rank function.

A.5 CONTINUOUS EXTENSIONS AND WHY ROUNDING IS NEEDED

For a set function, the multilinear extension is

$$F(x) = \mathbb{E}[f(R_x)], \qquad \mathbb{P}[e \in R_x] = x_e \text{ independently}, \quad x \in [0,1]^V.$$

Feasibility of $x \in P(M)$ does not make every independently sampled R_x feasible. For $U_{1,2}$ and $x = (1/2, 1/2)$, both elements are sampled with probability $1/4$. A rounding algorithm correlates the choices to produce an independent set while retaining expected objective value.

Lemma A.11 (Standard multilinear calculus)**.** *For a monotone submodular set function, $\partial_e F \ge 0$, $\partial^2_{ee} F = 0$, and $\partial^2_{ed} F \le 0$ for $e \ne d$. Consequently F is concave on every nonnegative ray and convex on every feasible exchange segment. In particular,*

$$F(x + \gamma(\mathbf{1}_v - \mathbf{1}_u)) \ge F(x) + \gamma(\partial_v F(x) - \partial_u F(x)).$$

Proof. Condition on all random memberships except e. Then $\partial_e F(x) = \mathbb{E}[f(R + e) - f(R)] \ge 0$ and the derivative is independent of x_e. Conditioning on all but e, d expresses the mixed derivative as an expectation of $f(R+e+d) - f(R+e) - f(R+d) + f(R)$, which is nonpositive by submodularity. For a nonnegative direction w, the second derivative is $w^\top \nabla^2 F w \le 0$. For direction $\mathbf{1}_v - \mathbf{1}_u$, it is $-2\partial^2_{uv} F \ge 0$. The displayed bound is the tangent inequality for a convex one-dimensional function. These are the classical analytic facts behind continuous greedy and pipage/swap rounding (Calinescu et al., 2011; Chekuri et al., 2010); the proof is included to fix signs and conventions. □

Example A.12 (Concavity and exchange convexity are different)**.** For two items covering the same unit-weight feature, $F(x_1, x_2) = x_1 + x_2 - x_1x_2$. Along (t, t) it is $2t - t^2$, concave. Along $(1/2 + s, 1/2 - s)$ it is $3/4 + s^2$, convex. Thus moving mass between coordinates can increase value even though increasing both coordinates has diminishing returns. Both properties will be used, with different roles.

A.6 A COMMON ROUNDING INPUT AND BOUNDARY CONVENTIONS

A rounder receives an explicit list

$$x = \sum_{i=1}^{t} \beta_i \mathbf{1}_{B_i}, \qquad \beta_i > 0, \quad \sum_i \beta_i = 1, \tag{A.1}$$

where the B_i are bases. It is not asked to discover such a decomposition from a general membership oracle. If independent sets must be padded, add r zero-value dummy elements and truncate $M \oplus U_{r,r}$ to rank r. An augmented set A is independent exactly when $|A| \le r$ and $A \cap V \in \mathcal{I}$; its rank is $\min\{r, r_M(A \cap V) + |A \setminus V|\}$. Each augmented query uses one original query. Delete dummies at the end. If $r = 0$, return the empty set; if a rounding input has $t = 1$, return that base. Loops can be deleted by singleton feasibility tests. These conventions also apply inside minors.

B RELATED WORK AND THE REMAINING COMPLEXITY GAP

The approximation target is classical; the question here is how cheaply it can be reached for an arbitrary oracle-given constraint. We distinguish three issues throughout this comparison: ordinary value queries versus stronger objective access, a general matroid oracle versus an explicit representation of a special matroid, and polynomial versus exponential dependence on $1/\varepsilon$. The literature comparison includes public papers available through 17 September 2026. The polymatroid construction proved in Section I is the contribution of this paper.

B.1 TWO LOWER BOUNDS: APPROXIMATION QUALITY AND QUERY COMPLEXITY

The first lower bound concerns *quality*. Even a cardinality constraint is a matroid constraint, and improving $1 - 1/e$ by a fixed positive constant requires exponentially many value queries in the worst case. The classical cardinality lower bound is due to Nemhauser–Wolsey (Nemhauser and Wolsey, 1978); Vondrák's symmetry-gap framework (Vondrák, 2013, Theorem 1.8 and Section 2) also covers randomized algorithms. Thus a $(1 - 1/e - \varepsilon)$ guarantee approaches the optimal polynomial-query approximation. This fact alone says nothing about whether the number of queries should be nearly linear or quadratic.

The second lower bound concerns *work at a constant approximation*. Peng–Rubinstein (Peng and Rubinstein, 2025, Theorem 3.7) prove an $\Omega(\alpha^5 n/\log^2 n)$ value-query lower bound for finding an α-approximate cardinality-constrained solution, in their range $r \le c\alpha n$ for a sufficiently small absolute constant c. Their hard objectives are already additive. Their Theorem 3.9 gives a corresponding $\Omega(\alpha^{11} n/\log^2 n)$ bound even for distinguishing two constant-factor-separated optimum values of monotone submodular functions, in the range $r \le c'\alpha^2 n$. In particular, for constant $\alpha > 0$ these are $\widetilde{\Omega}(n)$ bounds, including every regime $r = o(n)$.

Proposition B.1 (Consequence for our oracle models)**.** *Fix a constant $a \in (0, 1)$. A randomized algorithm that always outputs a feasible set and guarantees $\mathbb{E}f(S) \ge a\mathrm{OPT}$ for every monotone submodular objective and every matroid needs $\Omega(n/\log^2(2n))$ ordinary value queries in the worst case, also when the cost is measured in expectation and the matroid is supplied by a rank oracle.*

Proof. Use the uniform matroid $U_{r,n}$, with, for example, $r = \lfloor n/\log(2n) \rfloor$. Its independence and rank answers are known without any information about f: they are $|A| \le r$ and $\min\{|A|, r\}$. Thus constraint queries cannot help evade the value-query lower bound of (Peng and Rubinstein, 2025, Theorem 3.7).

To connect that lower bound with our expectation conventions, suppose the worst-case expected number of value queries is Q. For $\mathrm{OPT} > 0$, the random variable $X = f(S)/\mathrm{OPT}$ lies in $[0, 1]$ and

has expectation at least a. Consequently

$$\mathbb{P}[X \geq a/2] \geq \frac{a/2}{1 - a/2} \geq a/2.$$

Stop a run after $\lceil 4(Q+1)/a \rceil$ value queries and return the empty set if it has not finished. Markov's inequality bounds the probability of this interruption by $a/4$. The truncated run therefore finds an $(a/2)$-approximate solution with probability at least $a/4$. A constant number of independent runs, followed by value comparisons of their outputs, raises this to any fixed success probability below one. The total number of queries is $O_a(Q+1)$ on every run. Applying the cited lower bound at approximation $a/2$ gives the result. The case $\mathrm{OPT} = 0$ does not affect a worst-case lower bound. □

Our $n \operatorname{poly}(1/\varepsilon, \log(2n))$ upper bound therefore closes the *worst-case dependence on the ground-set size*, for fixed accuracy, to within polylogarithmic factors in both matroid oracle models. This matroid value-query comparison does not claim optimal dependence on $1/\varepsilon$ or a separate lower bound on constraint queries. It is also not a claim of optimality at every rank: for the free matroid $r = n$, monotonicity makes returning V optimal without evaluating f.

B.2 From accelerated continuous greedy to fast general matroids

Calinescu–Chekuri–Pál–Vondrák (Calinescu et al., 2011) established the continuous-greedy and rounding approach to the optimal approximation. Badanidiyuru–Vondrák, in *Fast algorithms for maximizing submodular functions* (Badanidiyuru and Vondrák, 2014, Theorem 1.3 and Section 4), made this approach much faster through decreasing thresholds, larger steps, and sampled marginal estimates. Their value-query bound is $O(nr\varepsilon^{-4}\log^2(n/\varepsilon))$; the generic rounding implementation still has a quadratic-in-rank term. Buchbinder–Feldman–Schwartz (Buchbinder et al., 2017) then introduced a preprocessing tradeoff that reduces subsequent sampling. Combined with classical swap rounding (Chekuri et al., 2010), it gives the $\widetilde{O}_\varepsilon(n\sqrt{r} + r^2)$ bound summarized in (Kobayashi and Terao, 2024, Section 1.1).

Ene–Nguyen's *Towards nearly-linear time algorithms for submodular maximization with a matroid constraint* (Ene and Nguyen, 2019) takes a different step: it reduces much of the work to maintaining a nearly maximum-weight base under weight decreases. For graphic and partition matroids, suitable data structures yield nearly linear algorithms. Henzinger–Liu–Vondrák–Zheng, in *Faster submodular maximization for several classes of matroids* (Henzinger et al., 2023), extend and improve this line for graphic, laminar, and transversal matroids. Their near-linearity is in the size of the supplied representation, which may exceed the number of matroid elements. These results do not provide the required data structures from an arbitrary independence oracle.

Kobayashi–Terao's *Subquadratic submodular maximization with a general matroid constraint* (Kobayashi and Terao, 2024, Theorems 1.1–1.3) removes the r^2 rounding bottleneck by searching for longer exchange cycles. It yields $\widetilde{O}(n\sqrt{r}\operatorname{poly}(1/\varepsilon))$ value and independence queries. With rank access, the authors implement the Ene–Nguyen base repairs efficiently and obtain $\widetilde{O}((n + r^{3/2})\operatorname{poly}(1/\varepsilon))$ queries. Their fractional rank algorithm is already nearly linear; the $r^{3/2}$ term comes from rounding.

A comparison at the same approximation target. Table 2 summarizes bounds for an integral $(1 - 1/e - \varepsilon)$-approximate solution. It records a common upper bound on the combined oracle counts, except for the two rows explicitly measuring time in a representation. All displayed $\widetilde{O}$ terms suppress logarithms; the accuracy column states whether further suppressed dependence is polynomial or exponential.

B.3 Local search, extended extensions, and recent developments

Buchbinder–Feldman's *Deterministic algorithm and faster algorithm for submodular maximization subject to a matroid constraint* (Buchbinder and Feldman, 2025a, Theorem 1.2) achieves the target approximation deterministically in $\widetilde{O}_\varepsilon(nr)$ queries and randomly in $\widetilde{O}_\varepsilon(n + r\sqrt{n})$ queries. The accuracy dependence is exponential: the auxiliary objective sums over subsets of $\ell = \Theta(1/\varepsilon)$ parts, so one evaluation uses 2^ℓ ordinary value queries (Buchbinder and Feldman, 2025a, Section 3.2, proof

Work	Constraint access	Bound	Accuracy cost
Badanidiyuru–Vondrák (Badanidiyuru and Vondrák, 2014)	General independence oracle	$\widetilde{O}(nr\ \mathrm{poly}(1/\varepsilon))$	Polynomial
Buchbinder–Feldman–Schwartz (Buchbinder et al., 2017)	General independence oracle	$\widetilde{O}((n\sqrt{r}+r^2)\,\mathrm{poly}(1/\varepsilon))$	Polynomial
Ene–Nguyen (Ene and Nguyen, 2019)	Explicit graphic or partition matroid	Nearly linear time in representation size	Polynomial
Henzinger et al. (Henzinger et al., 2023)	Explicit graphic, laminar, or transversal matroid	Nearly linear time in representation size	Polynomial
Kobayashi–Terao (Kobayashi and Terao, 2024)	General independence oracle	$\widetilde{O}(n\sqrt{r}\ \mathrm{poly}(1/\varepsilon))$	Polynomial
Kobayashi–Terao (Kobayashi and Terao, 2024)	General rank oracle	$\widetilde{O}((n+r^{3/2})\,\mathrm{poly}(1/\varepsilon))$	Polynomial
Buchbinder–Feldman, local search (Buchbinder and Feldman, 2025a)	General independence oracle	$\widetilde{O}_\varepsilon(n+r\sqrt{n})$	Exponential
Buchbinder–Feldman, extended extension (Buchbinder and Feldman, 2025b)	General independence oracle	$\widetilde{O}_\varepsilon(n+r^{3/2})$	Exponential
This paper	Either general matroid oracle	$n\,\mathrm{poly}(1/\varepsilon,\log(2n))$	Polynomial

Table 2: Selected near-optimal approximation algorithms. The two Buchbinder–Feldman rows use their randomized integral algorithms; their deterministic bounds are different.

of Theorem 1.2). It is nearly linear for sufficiently small rank at fixed accuracy, but the displayed bound remains $\widetilde{O}_\varepsilon(n^{3/2})$ when $r=\Theta(n)$.

Their subsequent STOC 2025 paper, *Extending the extension* (Buchbinder and Feldman, 2025b, Theorem 1.1 and Corollary 1.3), improves this general-matroid bound. It constructs a sparse extended-multilinear fractional solution using $O_\varepsilon(n\log r)$ queries. Combining this with Kobayashi–Terao rounding gives the monotone-case bound $\widetilde{O}_\varepsilon(n+r^{3/2})$. Evaluating the extended objective can use $2^{O(\varepsilon^{-4})}$ ordinary value queries; see its proof of Lemma 4.4 and concluding discussion. Thus even its nearly linear fractional phase does not provide a polynomial-accuracy nearly linear algorithm. Our new rounder alone would remove its $r^{3/2}$ cost at fixed accuracy, but would retain that exponential dependence. The new preprocessing and residual fractional algorithm are needed to obtain both properties together.

The 2026 Poisson-process papers offer a different algorithmic mechanism. Ganz Rozenman–Kulik–Schwartz–Singh (Ganz Rozenman et al., 2026, Theorems 1.1–1.3) maintain an integral base through randomly timed exchanges, avoiding a separate rounding stage. Their particularly fast ordinary-value result is for partition matroids. Kulik–Oliveira–Schwartz–Singh (Kulik et al., 2026, Theorems 1.1–1.3) extend the process to non-monotone objectives while retaining the monotone approximation. Their general ordinary-value implementation has an $\widetilde{O}(nr^2\varepsilon^{-2})$ bound, rather than a nearly linear bound for arbitrary rank. The number of Poisson events alone does not count the work required to estimate gradients and find exchanges. These event-driven processes are also different from the independent-coordinate Poisson smoothing that we use for integer counts.

Another recent result, Nematollahi–Vladu–Zhao (Nematollahi et al., 2026), reaches $1-1/e-\varepsilon$ even for non-monotone objectives in $n\,2^{\widetilde{O}(r/\,\mathrm{poly}(\varepsilon))}$ time. It treats rank as a fixed parameter. It does not give a nearly linear bound uniform over arbitrary rank. Taken together, the comparisons above leave the gap addressed here: ordinary value access, an arbitrary matroid supplied only by an oracle, an integral output, and polynomial accuracy dependence, all within nearly linear query complexity.

B.4 INTEGER-LATTICE OBJECTIVES AND POLYMATROID CONSTRAINTS

Soma–Yoshida's *Maximizing monotone submodular functions over the integer lattice* (Soma and Yoshida, 2018, Section 4) gives a polynomial-time $(1 - 1/e - \varepsilon)$ approximation for monotone DR-submodular objectives over an integral polymatroid. Their stated complexity is $\widetilde{O}(n^3\varepsilon^{-5}\log^2 r + n^8)$ in their unit-cost oracle model. They already batch count increments by binary search, use a continuous extension, and reduce local rounding to a matroid. The n^8 term is the exact decomposition used in rounding. Thus batching itself is not new here. Our algorithm combines a count-based finite-step analysis with approximate local covering, and controls the remaining numerical-rank costs through capacity scales, geometric search, and aggregate sampling errors. Section I gives its complete construction.

Ene–Nguyen (Ene and Nguyen, 2016) show how logarithmic encodings reduce lattice DR-submodular objectives to set objectives. Such an encoding does not by itself preserve a general polymatroid constraint as a matroid; the example in Section C explains why. Explicitly storing $\Theta(\log r)$ blocks per type is already quadratic in dimension at exponential rank. We instead process only occupied capacity groups. Random gaps between groups lose $O(\varepsilon)$ expected value and permit a nearly feasible product relaxation. A small integer baseline bounds within-group marginal ranges. All constraints still use one-bit integer feasibility, without rank-function or real-point membership access.

A count-based bound with a factor $\log^3 r$ would still contribute n^3 when $r = 2^{\Theta(n)}$. The theorem here displays every residual rank dependence through $\Xi_\varepsilon = \log(2n/\varepsilon) + \log\log(r + 3)$. The ordinary-value and expected feasibility bounds have accuracy powers nine and eleven respectively. Optimal accuracy dependence remains open.

Because matroids are special integral polymatroids, Proposition B.1 supplies a worst-case value-query lower bound in the number of types. Formally, extend a set objective to integer vectors by $g(x) = f(\{e : x_e \geq 1\})$ and use $P(M)$ as the polymatroid. This extension is monotone DR-submodular, one lattice value query is simulated by one set-value query, and feasible integer points are exactly independent-set indicators.

For the exponential-rank regime itself, Proposition I.26 gives a separate elementary $\Omega(n)$ expected feasibility-query lower bound at rank exactly 2^n, with a known linear objective. Its information barrier is identifying which of $n - 1$ coordinates admits one valuable unit; an independent zero-value coordinate supplies the exponential rank. Combined with the upper bound, this closes worst-case dimension dependence up to polylogarithmic factors at fixed accuracy. It does not prove that large rank itself increases query complexity, nor does it identify the optimal dependence on $1/\varepsilon$ or r. Query complexity also leaves the cost of representing large counts and dense query vectors outside the bound.

B.5 WHY BOTH MATROID ORACLE MODELS ARE RELEVANT

The models and their history are explained in Section A.3. Their difference was not absent from all earlier submodular-maximization research: Balkanski–Rubinstein–Singer (Balkanski et al., 2019, Section 4) already distinguish them when counting parallel rounds. In the sequential fast-algorithm line, the emphasis was often elsewhere. Badanidiyuru–Vondrák emphasize objective evaluations; Buchbinder–Feldman–Schwartz use the standard independence model; Ene–Nguyen and Henzinger et al. exploit explicit representations and dynamic base data structures. Kobayashi–Terao make the distinction central to the sequential query bounds by showing that rank queries implement the needed base repairs cheaply. Polynomial-time intersimulation of two oracles does not preserve nearly linear complexity.

There is a precise sense in which our independence result subsumes the *existence* of a nearly linear rank-oracle algorithm: a rank query answers every independence test. We nevertheless retain the direct rank algorithm because it reaches that guarantee through a much simpler route. It combines an existing fractional algorithm with fixed-length exchange walks; it needs neither multilevel private-core preprocessing nor the geometric stopping argument required by our independence implementation. It can therefore be used on its own when rank computation or dynamic base maintenance is already available.

A setting with native rank information. Suppose items are represented by vectors and the feasible selections must be linearly independent, while a coverage objective measures their utility. A linear-algebra service may report the dimension of the span of any queried collection; that is exactly a matroid rank oracle. For an independent set J of size $r-1$, the query $r_M(J \cup T) > |J|$ answers whether an entire group T contains a valid replacement. Repeating this test on halves finds a replacement in $O(\log(2|T|))$ service calls. A Boolean independence response for the same oversized collection would discard this information. Similarly, for a graphic matroid, a connected-components routine returns rank as $|W| - c(W, A)$ on an edge set A of a graph with vertex set W. These are settings where the rank algorithm uses information a natural constraint routine already computes. They are not assertions that such routines take constant arithmetic time, or that the rank algorithm is always faster in practice. Our theorem measures service calls; actual implementation cost depends on the representation.

C Main results and technical overview

C.1 Results and the order of presentation

For set objectives let $\mathrm{OPT} = \max_{I \in \mathcal{I}} f(I)$, and for lattice objectives let $\mathrm{OPT} = \max_{x \in P(\rho) \cap \mathbb{Z}^E} f(x)$. All final approximation guarantees below are in expectation; outputs are feasible on every run. By nearly linear dependence on dimension we mean $n \operatorname{poly}(1/\varepsilon, \log(2n))$ oracle calls. For polymatroids, a factor $\log^3 r$ would violate this target when $r = 2^{\Theta(n)}$. We therefore display numerical-rank dependence explicitly: our refined bound depends on $\log\log(r+3)$, which is $O(\log(2n))$ throughout that regime. The accuracy dependence remains polynomial.

Theorem C.1 (Unified approximation guarantees)**.** *For $0 < \varepsilon \le 1/2$, normalized nonnegative monotone objectives admit the following algorithms.*

(i) A submodular set function under an arbitrary rank-oracle matroid admits a $(1 - 1/e - \varepsilon)$ approximation in $O(n \operatorname{poly}(1/\varepsilon, \log(2n)))$ value and rank queries.

(ii) Under an arbitrary independence-oracle matroid, the same guarantee holds with expected query count

$$O(n \operatorname{poly}(1/\varepsilon, \log(2n)))$$

for value and independence access.

(iii) A monotone DR-submodular lattice function over an arbitrary integral polymatroid admits that guarantee in $O(n\varepsilon^{-9}\Xi_\varepsilon^3)$ value queries and $O(n\varepsilon^{-11}\Xi_\varepsilon^3)$ expected integer-feasibility queries, where $\Xi_\varepsilon = \log(2n/\varepsilon) + \log\log(r+3)$. In particular, both bounds are $n \operatorname{poly}(1/\varepsilon, \log(2n))$ when $r = 2^{\Theta(n)}$.

The set-function results first use fractional solutions with $t = O(1/\varepsilon)$ bases. Their new rounding bounds, stated for an arbitrary supplied list of t bases, are

$$Q_{\mathrm{rank}} = O(rt^2\eta^{-1}\log^2(2r)), \qquad \mathbb{E}Q_{\mathrm{ind}} = O(r(t-1)^2\eta^{-1}\log(2r)),$$

with expected value at least $(1-\eta)F(x)$ and no value queries. For polymatroids the local rounder uses $O(nD^2\eta^{-5}\log^3(2n))$ expected feasibility queries at denominator D. Our fractional construction and final feasibility-preserving shrink give $D = O(\varepsilon^{-3})$. Theorem I.1 proves item (iii). Proposition I.26 gives an $\Omega(n)$ expected feasibility lower bound at rank exactly 2^n, even for a known linear objective. Thus the dimension dependence is tight up to polylogarithmic factors at fixed accuracy. No optimality claim is made for the accuracy or doubly logarithmic rank dependence.

Published ingredients and proofs. The rank-oracle fractional phase is imported from published work. The walk rounders, multilevel preprocessing, and polymatroid construction are proved in full below. Published subroutines have explicit guarantees and source locations. The polymatroid proof presents one algorithm with doubly logarithmic rank dependence.

C.2 The route to nearly linear complexity

We begin with rounding because it exposes the main change most simply. With rank access, a short walk and a random starting point suffice. With independence access, the searchable exchange

direction reverses, so the same argument charges the wrong endpoint. Geometric stopping repairs that mismatch by charging elements actually removed, whose total charge has one global bound. These mechanisms are developed in Sections D–F.

The fractional problem needs a different saving. Preprocessing already makes continuous-greedy sampling cheap, but repeatedly searching and rebuilding bases is expensive. Our preprocessing ensures that each such expense either certifies many *different* low-value elements or moves to a smaller matroid where the same task is cheaper. Repeating this improvement over several levels makes its exponent approach one (Section G). Section H supplies the continuous-greedy implementation and the composition proof; its role is to use the improved preprocessing guarantee.

For polymatroids, a coordinate can represent exponentially many units. Section I presents the final algorithm directly. Capacity-scale separation permits a nearly feasible product relaxation; an integer baseline controls each group's marginal range. Poisson smoothing and colors supply large, valuable steps within the residual problem. Geometric count searches and aggregate concentration keep both tests and samples inexpensive. A final shrink returns to the original constraint, and denominator-aware coverage makes rounding nearly linear. Shadow counts justify the search comparison only in the proof; every implemented direction uses real feasible counts.

C.3 The two bottlenecks in the previous framework

The usual route is to find a valuable fractional point and then round it. These are separate computational problems. A faster fractional algorithm does not help if rounding is still expensive; a faster rounder does not remove the sampling cost of finding its input.

The following two published guarantees describe the independence-oracle framework that we improve. The first returns a short weighted list of bases; the second converts that list into one base. Their implementations are used here only for comparison, so we state their guarantees rather than reproduce their algorithms. Let $0 < \tau < 1/2$ be the accuracy and $1 \le \lambda \le r$ the preprocessing tradeoff parameter. Rank zero is trivial and is handled separately.

Proposition C.2 (Published fractional solution)**.** *The Buchbinder–Feldman–Schwartz construction, in the fractional form of (Kobayashi and Terao, 2024, Theorem 3.1 and Remark 3.2), returns $t = O(1/\tau)$ explicit bases and coefficients summing to one, with*

$$\mathbb{E}F\left(\sum_i \beta_i \mathbf{1}_{B_i}\right) \ge (1 - 1/e - \tau)\mathrm{OPT}.$$

Its value-query cost is $\widetilde{O}(r\lambda + rn/\lambda)$ and its independence-query cost is $\widetilde{O}(n\lambda)$, suppressing polynomial accuracy dependence. The construction is in (Buchbinder et al., 2017, Section 3.1, Algorithms 1–2, and Lemma 3.4).

Its preliminary greedy stage selects a prefix and reduces the ranges of remaining marginal samples. Increasing λ spends more on this preprocessing and less on the residual continuous-greedy stage.

The rounding call receives only the matroid, base list, and allowed loss.

Proposition C.3 (Kobayashi–Terao rounding)**.** *For an explicit base list $x = \sum_{i=1}^{t} \beta_i \mathbf{1}_{B_i}$ and $0 < \eta < 1$,* **KobayashiTeraoRound** *returns a base R satisfying $\mathbb{E}[f(R) \mid x] \ge (1 - \eta)F(x)$. It uses $O(r^{3/2} t \log^{3/2}(rt/\eta))$ independence queries and no value queries (Kobayashi and Terao, 2024, Theorem 1.2).*

Taking $\tau = \varepsilon/3$ in both calls gives expected value at least $(1 - \tau)(1 - 1/e - \tau)\mathrm{OPT} \ge (1 - 1/e - \varepsilon)\mathrm{OPT}$. We use this guarantee without reproducing the rounder's cycle-search algorithm.

The two remaining costs. Because $r \le n$, the dominant fractional terms are $n\lambda$ and nr/λ. Their balance at $\lambda = \sqrt{r}$ gives $\widetilde{O}(n\sqrt{r})$. Rounding separately costs $\widetilde{O}(r^{3/2})$ for fixed accuracy. Improving only the rounder therefore leaves the fractional bottleneck in place.

Example C.4 (A square-root balance)**.** With $n = 10^6$ and $r = 10^4$, choosing $\lambda = 10, 100, 1000$ gives $(n\lambda, nr/\lambda) = (10^7, 10^9), (10^8, 10^8), (10^9, 10^7)$. Even the balanced choice is a factor 100 above n in these cost terms. These are asymptotic expressions with logarithms and accuracy omitted, not measured running times.

The rounding bottleneck has a different cause. Classical swap rounding (Chekuri et al., 2010) finds a pair of mutually feasible exchanges; this is a cycle of length two. Kobayashi–Terao (Kobayashi and Terao, 2024, Section 5) permit longer cycles, which are cheaper to find. Around a cycle, the fractional displacements cancel. Exchange-direction convexity then guarantees no expected loss from the cycle update. The direct pair-search implementation costs $O(r^2t)$ queries, and the long-cycle method reduces the bound to the one above.

Our change is to stop insisting that the walk closes. The interior terms still cancel, and we pay only for the endpoints. This trades exact cancellation for an $O(\varepsilon)$ loss. The important question becomes which endpoint has the unfavorable sign and how its loss is shared across all updates. Rank and independence access lead to different answers. This comparison concerns the swap/cycle line of rounding methods; it is not a characterization of every previous rounding algorithm.

C.4 Why continuous greedy gives the approximation constant

The continuous-greedy framework of Calinescu, Chekuri, Pál, and Vondrák (Calinescu et al., 2011) constructs a fractional point, then rounds it. Its idealized calculation explains the approximation constant. Starting at $x(0) = 0$, choose

$$v(t) \in \arg\max_{v \in P(M)} v \cdot \nabla F(x(t)), \qquad x'(t) = v(t).$$

Since an optimum indicator $\mathbf{1}_O$ is feasible, monotonicity and continuous DR imply

$$v(t) \cdot \nabla F(x(t)) \geq \sum_{e \in O} \partial_e F(x(t)) \geq F(x(t) \vee \mathbf{1}_O) - F(x(t)) \geq \mathrm{OPT} - F(x(t)).$$

The middle inequality follows by adding the missing coordinates of O one at a time and bounding each increase by its initial derivative. Thus $F'(x(t)) \geq \mathrm{OPT} - F(x(t))$, and integrating gives $F(x(1)) \geq (1 - e^{-1})\mathrm{OPT}$. The notation $F'(x(t))$ here means the derivative of the scalar trajectory $t \mapsto F(x(t))$.

Three implementation costs are absent from this differential inequality: sampling gradients through an ordinary value oracle, constructing feasible weighted directions through the available constraint oracle, and rounding the resulting fractional point. Nearly linear fractional optimization alone does not settle the integral problem.

C.5 What rank access already supplies

Ene–Nguyen (Ene and Nguyen, 2019, Algorithms 1–2 and Lemmas 7–14) obtain fast fractional optimization when a maximum-weight base can be maintained under single-element weight decreases. Kobayashi–Terao (Kobayashi and Terao, 2024, Lemmas 6.1–6.3) implement that dynamic operation by rank-query binary search. Their complete rank-oracle algorithm still has the $n + r^{3/2}$ bound because of rounding (Kobayashi and Terao, 2024, Theorem 1.3). We therefore begin with rank access: its fractional algorithm is already available, and the remaining proof is a particularly transparent walk analysis.

C.6 The integer-lattice predecessor and its square-root balance

Soma–Yoshida (Soma and Yoshida, 2018, Sections 4.1 and 4.2) already use binary-search batching and a continuous extension formed by multilinear interpolation within each unit cube. Their local rounding reduction leads to a matroid; the exact convex-decomposition step in their swap-rounding implementation costs $O(n^8)$ (Soma and Yoshida, 2018, Section 4.2.2). The challenge here is to make both the fractional stage and that local rounding step nearly linear. Ene–Nguyen (Ene and Nguyen, 2016) give logarithmic block encodings for lattice objectives. A compact block encoding by itself need not preserve matroid augmentation, as the following example explains.

Simply replacing each type by its u_e units creates up to nr elements. Applying a nearly linear matroid algorithm to that expansion would be polynomial in numerical rank. Conversely, a small block encoding does not make its feasible block subsets a matroid.

Example C.5 (A logarithmic encoding need not preserve augmentation)**.** For $x_1 + x_2 + x_3 \leq 3$, encode each coordinate by blocks of sizes one and two. The block sets $A = \{b_{1,2}, b_{2,1}\}$ and

$B = \{b_{1,1}, b_{2,1}, b_{3,1}\}$ both have total weight three. Although $|A| = 2 < 3 = |B|$, neither element of $B \setminus A$ can augment A. Thus the block constraint is not a matroid. Our polymatroid algorithm uses counts for implementation and unit-token matroids only for analysis.

D RANK ACCESS: THE PUBLISHED NEARLY LINEAR FRACTIONAL PHASE

D.1 THE FRACTIONAL SUBROUTINE

With a rank oracle, the fractional problem already has a nearly linear solution. Given M, f, and an accuracy, the published algorithm returns an explicit short list of bases. It first selects a prefix that reduces remaining marginal ranges, then applies fast continuous greedy to the residual problem. Its efficient implementation maintains a maximum-weight base as individual estimated weights decrease. Rank queries make each such repair a binary search.

Theorem D.1 (Published fractional algorithm)**.** *Given a nonnegative monotone submodular function, a rank-r matroid, and $0 < \xi < 1$, the framework in (Kobayashi and Terao, 2024, Lemma 6.1) returns*

$$x = \sum_{i=1}^{t} \beta_i \mathbf{1}_{B_i}, \quad t = O(1/\xi), \quad \mathbb{E}[F(x)] \geq (1 - 1/e - \xi)\mathrm{OPT}.$$

The B_i are explicit bases and the coefficients are positive and sum to one. Its operation count is

$$O(n \operatorname{poly}(1/\xi, \log(2n))).$$

The counted operations are ordinary value queries, independence tests, and maximum-weight-base repairs after single-element weight decreases. We call this algorithm **RankFractional** *when the latter operations are implemented by Lemmas D.2–D.3.*

The saving: repair one position. A decreased marginal estimate does not require rebuilding an entire base. If its element is in the maintained base, only one replacement is needed; otherwise the base remains optimal. The two lemmas below implement this replacement in $O(\log(2n))$ rank queries. Multiplying the theorem's operation count by this logarithm preserves near-linearity.

Theorem D.1 is (Kobayashi and Terao, 2024, Lemma 6.1), consolidating (Ene and Nguyen, 2019, Algorithms 1–2 and Lemmas 7–14). Lemmas 6.2–6.3 of (Kobayashi and Terao, 2024) give the rank implementation. We use the conservative polynomial accuracy bound common to those statements. Specifically, the source's Lemma 6.1 records $O(n\xi^{-5}\log^2(n/\xi))$ value queries, $O(n\xi^{-1}\log(n/\xi))$ independence tests, and $O(r\xi^{-1}\log^2(n/\xi))$ base-maintenance operations. We only need their common $n \operatorname{poly}(1/\xi, \log(2n))$ bound and the explicit base list. This avoids relying on finer accuracy and refresh counts that differ between statements and analyses in the Ene–Nguyen source.

D.2 WEIGHTED RANK SEARCH AND A ONE-WEIGHT REPAIR

Lemma D.2 (Published weighted extension search: **RankExtend**)**.** *Given an independent set J and candidates $T \subseteq V \setminus J$, one can find a maximum-weight $u \in T$ such that $J + u$ is independent, or determine that none exists, using $O(\log(2|T|))$ rank queries. We denote the returned element by* **RankExtend**(J, T)*, with weights omitted for an unweighted search.*

Proof. This is the binary-search primitive in (Kobayashi and Terao, 2024, Lemma 6.2). First test whether $r_M(J \cup T) > |J|$. A negative answer means all candidates are in $\mathrm{cl}(J)$. Otherwise sort candidates by weight, using fixed tie-breaking. Test the higher-weight half through the same rank predicate. If it contains an extension, keep it; otherwise keep the lower half. The existence promise is preserved because the candidates in a failed half all lie in $\mathrm{cl}(J)$. After logarithmically many tests a singleton remains. Ordering explicit weights is internal computation; no objective query is used. □

Lemma D.3 (Single-weight sensitivity)**.** *Let B maximize a weight vector w, and suppose only $w(v)$ decreases. If $v \notin B$, B remains optimal. If $v \in B$, a maximum-weight extension of $B - v$ under the new weights is an optimal new base.*

Proof. This is the standard sensitivity fact used in (Kobayashi and Terao, 2024, Lemma 6.3). For completeness, write w' for the changed vector and let B' be the best extension of $J = B - v$. If $v \notin B$, every competing base either loses weight or remains unchanged, so the first assertion follows. Assume $v \in B$. For any competing base D containing v, both B and D lose the same amount, so $w'(B) \geq w'(D)$ and $w'(B') \geq w'(B)$. If $v \notin D$, the symmetric basis-exchange theorem supplies $u \in D \setminus B$ with both $B - v + u$ and $D - u + v$ bases. Old optimality gives $w(B) \geq w(D) - w(u) + w(v)$, hence $w'(B - v + u) = w(B) - w(v) + w(u) \geq w'(D)$. The maximum-weight extension B' is at least as good. The symmetric exchange fact is classical matroid theory (Oxley, 2011); the argument here only specializes it to one decreasing weight. □

The candidate set for a repair is $(V \setminus B) \cup \{v\}$, so Lemma D.2 performs it in $O(\log(2n))$ rank queries. An initial maximum-weight base is found by sorting and greedy, with at most n independence tests, each simulated by one rank query.

Example D.4 (Repair instead of rebuilding)**.** In $U_{2,4}$, let the weights of a, b, c, d be $9, 8, 7, 2$. The base $\{a, b\}$ is optimal. If b's weight drops to one, keep $J = \{a\}$ and search candidates c, d, b in that order; c is the highest-weight valid extension, giving $\{a, c\}$. For a general matroid the highest-weight candidate may be blocked, but rank search discards whole candidate groups. An independence-only query on J plus a large candidate group would not implement this search.

D.3 Why the fractional representation stays small

The preconditioner returns an independent prefix S and a residual problem on M/S. If the continuous phase uses directions $I_1, \ldots, I_t$ with step sizes summing to one, it constructs their convex combination online. Extend directions to bases when necessary. Then

$$\mathbf{1}_S + \sum_i \beta_i \mathbf{1}_{B_i} = \sum_i \beta_i \mathbf{1}_{S \cup B_i}$$

is an explicit decomposition into original-matroid bases. It is not the output of a separate, potentially expensive polyhedral decomposition procedure. This representation is part of the published guarantee and is precisely what the new rounding algorithms consume.

Combining Theorem D.1 with the rank implementation above already gives a nearly linear fractional algorithm. The next section supplies the missing nearly linear integral step.

E Rank access: fixed-length alternating-walk rounding

E.1 Open the cycle and pay for one endpoint

Suppose two bases disagree. A swap-rounding update would search for exchanges that close into a cycle, so that their average displacement is zero. Instead, build a short chain of one-sided exchanges and stop wherever it leads. Choose one link uniformly and perform one of its two exchanges. Every outcome remains feasible and removes one disagreement. Only the *expected objective value* needs an approximate guarantee.

For the moment, think of each element as carrying its current marginal value. At every interior vertex, the walk contributes that value once with a plus sign and once with a minus sign. The average change therefore contains only

$$\frac{c}{L}\big(\text{marginal at the end} - \text{marginal at the start}\big),$$

where L is the number of links and $c \leq 1$ is determined by the two base coefficients. Derivatives make this informal calculation rigorous below. Monotonicity lets us drop the favorable ending term. Thus we need to control only the starting element, which we are free to choose uniformly among the d disagreements.

There are two savings, and both matter. Averaging the links contributes $1/L$; the uniform start contributes $1/d$. Across a merge, the disagreement size falls from at most r to zero, so the losses add as $L^{-1}(1 + 1/2 + \cdots + 1/r)$, rather than r/L. For a short base list, a logarithmic-length walk is

enough at fixed accuracy. Rank queries find each link by binary search, giving nearly linear work overall. Walks may repeat vertices; no simple-path or cycle-search requirement remains.

The published fractional phase from Section D, run with accuracy $\varepsilon/2$, supplies the list. Rounding it with allowed loss $\varepsilon/2$ proves the following result. We now give the new rounding algorithm and its full analysis.

Theorem E.1 (Nearly linear rank-oracle maximization)**.** *A value oracle and a rank oracle suffice for an expected* $(1-1/e-\varepsilon)$ *approximation with* $O(n\,\mathrm{poly}(1/\varepsilon, \log(2n)))$ *queries. With an* $O(1/\varepsilon)$*-base input, rounding uses* $O(r\varepsilon^{-3}\log^2(2r))$ *rank queries and no value queries.*

E.2 The fixed-length alternating walk

Suppose one update has two bases B_1, B_2 with coefficients $\alpha, \beta > 0$. Write

$$A = B_1 \setminus B_2, \qquad C = B_2 \setminus B_1, \qquad d = |A| = |C|. \tag{E.1}$$

If $d = 0$, the two bases already coincide and their coefficients are simply added. Assume $d > 0$ and fix an integer $L \geq 1$.

E.2.1 One update

The update receives these two bases and a length L, and returns the bases after one exchange. It first builds the whole walk in the unchanged bases, then chooses a link uniformly and executes one of that link's two exchanges. Repeated vertices are allowed.

Algorithm E.1. RankWalkUpdate$(B_1, B_2, \alpha, \beta, L)$

$A \leftarrow B_1 \setminus B_2$; $C \leftarrow B_2 \setminus B_1$. Choose u_0 uniformly from A.
for $i = 0, \ldots, L-1$ **do**
  $v_i \leftarrow$ **RankExtend**$(B_1 - u_i, C)$.
  $u_{i+1} \leftarrow$ **RankExtend**$(B_2 - v_i, A)$.
end for
Choose I uniformly from $\{0, \ldots, L-1\}$.
if an independent coin of success probability $\beta/(\alpha+\beta)$ succeeds **then**

$$B_1 \leftarrow B_1 - u_I + v_I. \tag{E.2}$$

else

$$B_2 \leftarrow B_2 - v_I + u_{I+1}. \tag{E.3}$$

end if
return (B_1, B_2).

The two searches guarantee, in the frozen bases,

$$B_1 - u_i + v_i \text{ and } B_2 - v_i + u_{i+1} \text{ are bases.} \tag{E.4}$$

Averaging over the link cancels its intermediate derivative; averaging across the walk cancels all interior endpoints. Only the starting and ending derivatives remain. The next lemmas bound the search cost and the single unfavorable endpoint.

Every possible outcome is a base by construction. Moreover, in (E.2), the inserted element v_I already belongs to B_2, while the removed element u_I does not; hence the new disagreement size is $d-1$. The same argument applies to (E.3). Thus each update makes definite combinatorial progress.

Example E.2 (A three-link walk with a nonzero bias)**.** Use $U_{2,4}$, $B_1 = \{a, b\}$, $B_2 = \{c, d\}$, and $\alpha = \beta = 1/2$. Let the modular weights be $(w_a, w_b, w_c, w_d) = (1, 7, 4, 2)$, so $F(x) = 7$ initially. Take walk length $L = 3$. Choose each entrant in $B_2 \setminus B_1$ to be c and each following vertex in $B_1 \setminus B_2$ to be a. If $u_0 = a$, the endpoint difference is zero. If $u_0 = b$, it is $1 - 7 = -6$. Since u_0 is uniform, the mean value change is

$$\frac{\alpha\beta}{(\alpha+\beta)L} \cdot \frac{0-6}{2} = -\frac{1}{4}.$$

For a modular objective the tangent inequalities are equalities, so the expected value after the update is exactly 6.75. All outcomes are bases. The proof's lower bound is $(1-1/(Ld))F(x) = (1-1/6)7$. The example shows both the presence of bias and why exact mean preservation is unnecessary for a small approximation loss.

E.2.2 Rank queries generate the walk

The following group test is the reason that a long walk is inexpensive.

Lemma E.3 (Rank binary search)**.** *Let J be independent with $|J| = r-1$, and let $T \subseteq V \setminus J$ satisfy $\mathrm{rank}(J \cup T) = r$. An element $e \in T$ such that $J + e$ is a base can be found using at most $\lceil \log_2 |T| \rceil$ rank queries.*

Proof. Maintain a candidate set T' with the promise $\mathrm{rank}(J \cup T') = r$. If $|T'| = 1$, its only element is valid. Otherwise split $T' = T_1 \dot\cup T_2$ into two parts whose sizes differ by at most one and query $\mathrm{rank}(J \cup T_1)$.

If the answer is r, retain T_1. Otherwise the answer is $r-1$, because J already has that rank. All elements of T_1 then lie in the closure of J. Since $J \cup (T_1 \cup T_2)$ has rank r, the set T_2 must contain an element outside that closure; retain T_2. The promise is preserved and the candidate set is at least halved. After at most $\lceil \log_2 |T| \rceil$ queries a singleton remains. □

To find v_i, apply Lemma E.3 with $J = B_1 - u_i$ and $T = C$. The promise holds because $(B_1 - u_i) \cup C$ contains B_2. To find u_{i+1}, use $J = B_2 - v_i$ and $T = A$; now the queried union contains B_1. Hence a length-L walk costs at most

$$2L\lceil \log_2(2r) \rceil \tag{E.5}$$

rank queries. No strongly exchangeable pair and no simple cycle is needed.

E.3 Why the endpoint bias is affordable

At the moment of an update, the full fractional point is

$$x = \alpha \mathbf{1}_{B_1} + \beta \mathbf{1}_{B_2} + z, \tag{E.6}$$

where $z \geq 0$ is the contribution of all bases not currently being merged. All derivatives below are evaluated at this current x.

Why does a uniform start contribute $1/d$? All d possible starting coordinates carry at least α fractional mass. Their combined marginal value cannot exceed $F(x)/\alpha$: deleting that much mass would already pay for it. A uniform start receives only a $1/d$ share of this total. The following lemma formalizes this observation.

Lemma E.4 (Derivative mass)**.** *Let $Q \subseteq V$ and suppose $x_u \geq a > 0$ for every $u \in Q$. Then*

$$a \sum_{u \in Q} \partial_u F(x) \leq F(x). \tag{E.7}$$

Proof. Put $y = x - a\mathbf{1}_Q \geq 0$ and define $h(s) = F(y + s\mathbf{1}_Q)$ for $0 \leq s \leq a$. Submodularity of f implies that the multilinear extension is concave along every nonnegative direction, so h is concave. Therefore its derivative at the right endpoint is at most its average slope:

$$ah'(a) \leq h(a) - h(0). \tag{E.8}$$

Here $h(a) = F(x)$, while $h(0) = F(y) \geq 0$ because f is nonnegative. Finally,

$$h'(a) = \sum_{u \in Q} \partial_u F(x). \tag{E.9}$$

Substitution into (E.8) proves (E.7). □

Example E.5 (The derivative-mass estimate)**.** For two duplicate coverage items at $x = (1/2, 1/2)$, $F(x) = 3/4$ and both derivatives are $1/2$. Taking $Q = \{1, 2\}$ and $a = 1/2$ gives $a \sum_{u \in Q} \partial_u F(x) = 1/2 \leq 3/4$. The deleted starting point in the proof is $(0, 0)$; the concave ray has value $2s - s^2$, and its final slope is below its average slope. The argument is valid for every nonnegative monotone submodular objective, not just coverage or modular examples.

Lemma E.6 (Loss in one update)**.** *Conditioned on the current point and bases, one alternating-walk update satisfies*

$$\mathbb{E}[F(x') \mid x] \geq \left(1 - \frac{1}{Ld}\right) F(x). \tag{E.10}$$

Proof. First condition on u_0 and on the entire constructed walk. If the first branch executes edge i, then

$$x' = x + \alpha(\mathbf{1}_{v_i} - \mathbf{1}_{u_i}); \tag{E.11}$$

if the second branch executes that edge, then

$$x' = x + \beta(\mathbf{1}_{u_{i+1}} - \mathbf{1}_{v_i}). \tag{E.12}$$

Both segments lie in the hypercube. Put

$$c = \frac{\alpha\beta}{\alpha + \beta}. \tag{E.13}$$

The branch probabilities make the coefficient of each of the two tangent directions equal to c. Applying Lemma A.11 to (E.11) and (E.12), and then averaging the uniformly selected index, gives

$$\begin{aligned}\mathbb{E}[F(x') \mid x, u_0, \text{walk}] &\geq F(x) + \frac{c}{L}\sum_{i=0}^{L-1}\left[\partial_{v_i}F(x) - \partial_{u_i}F(x) + \partial_{u_{i+1}}F(x) - \partial_{v_i}F(x)\right] \\ &= F(x) + \frac{c}{L}\left(\partial_{u_L}F(x) - \partial_{u_0}F(x)\right).\end{aligned} \tag{E.14}$$

Every internal derivative cancels. This cancellation is pointwise for every constructed walk, so repeated vertices and arbitrary tie-breaking in the rank searches cause no problem.

Monotonicity gives $\partial_{u_L}F(x) \geq 0$. Since u_0 is uniform in A, every $u \in A$ has $x_u \geq \alpha$, and $|A| = d$, Lemma E.4 gives

$$\mathbb{E}_{u_0}[\partial_{u_0}F(x)] = \frac{1}{d}\sum_{u \in A}\partial_u F(x) \leq \frac{F(x)}{\alpha d}. \tag{E.15}$$

Taking the remaining expectation in (E.14) and using (E.13) therefore yields

$$\begin{aligned}\mathbb{E}[F(x') \mid x] &\geq F(x) - \frac{c}{L}\frac{F(x)}{\alpha d} \\ &= \left(1 - \frac{\beta}{\alpha + \beta}\frac{1}{Ld}\right)F(x) \geq \left(1 - \frac{1}{Ld}\right)F(x),\end{aligned} \tag{E.16}$$

which is (E.10). □

Controlled drift. The fractional point is not a martingale. Its conditional mean drift is $\frac{c}{L}(\mathbf{1}_{u_L} - \mathbf{1}_{u_0})$. The proof instead shows directly that the multilinear value loses little: the walk cancels all internal drift, and the derivative-mass lemma pays for the single unfavorable endpoint.

E.4 The rounder and its complexity

The one-update argument is now complete. To round a weighted list, merge its bases one at a time. Each merge repeatedly calls **RankWalkUpdate** until its two bases agree, then adds their coefficients. The input is the explicit list (A.1); the output is one base.

Algorithm E.2. RankRound$((\beta_i, B_i)_{i=1}^t, \eta)$

if $t = 1$ or $r = 0$ **then**
  return B_1.
end if
$L \leftarrow \lceil (t-1)H_r/\eta \rceil$, where $H_r = \sum_{j=1}^r 1/j$.
$B \leftarrow B_1$; $\alpha \leftarrow \beta_1$.
for $j = 2, \dots, t$ **do**
  $C \leftarrow B_j$; $\beta \leftarrow \beta_j$.
  while $B \neq C$ **do**
    $(B, C) \leftarrow$ **RankWalkUpdate**(B, C, α, β, L).
  end while
  $\alpha \leftarrow \alpha + \beta$.
end for
return B.

Each update removes one disagreement. The bound $1/(Ld)$ just proved therefore sums to H_r/L per merge. Choosing L as above shares the allowed loss η among all $t-1$ merges. The following theorem makes this accounting precise.

Theorem E.7 (Alternating-walk rounding)**.** *Given* (A.1) *and* $0 < \eta < 1$*, the algorithm returns a base* S *satisfying*

$$\mathbb{E}[f(S)] \geq (1-\eta)F(x). \tag{E.17}$$

It makes no value query and uses

$$O\left(\frac{rt^2}{\eta}\log^2(2r)\right) \tag{E.18}$$

rank queries.

Proof. Consider one merge whose initial disagreement size is d_0. Every update reduces d by exactly one. Iterating Lemma E.6 with conditional expectations therefore gives

$$\begin{aligned}\mathbb{E}[F(x^{\text{after}}) \mid x^{\text{before}}] &\geq \prod_{j=1}^{d_0}\left(1-\frac{1}{Lj}\right)F(x^{\text{before}}) \\ &\geq \left(1-\frac{H_{d_0}}{L}\right)F(x^{\text{before}}) \geq \left(1-\frac{H_r}{L}\right)F(x^{\text{before}}),\end{aligned} \tag{E.19}$$

where $H_q = \sum_{j=1}^q 1/j$ and $\prod_j(1-a_j) \geq 1 - \sum_j a_j$ for $a_j \in [0,1]$.

The same statement applies conditionally when a base entering a later merge is random. Across the $t-1$ merges, another application of the product inequality gives

$$\mathbb{E}[f(S)] \geq \left(1-\frac{(t-1)H_r}{L}\right)F(x). \tag{E.20}$$

For $t > 1$, choose

$$L = \left\lceil \frac{(t-1)H_r}{\eta} \right\rceil; \tag{E.21}$$

if $t = 1$, return the given base. Equation (E.20) proves (E.17).

There are at most r updates per merge and $t-1$ merges. By (E.5), the total number of rank queries is at most

$$2r(t-1)\left\lceil \frac{(t-1)H_r}{\eta} \right\rceil \lceil \log_2(2r) \rceil. \tag{E.22}$$

Since $H_r \leq 1 + \log r$, this is $O(rt^2\eta^{-1}\log^2(2r))$. The construction never queries f. □

E.5 END-TO-END COMPOSITION

Proof of Theorem E.1. Run Theorem D.1 with $\delta = \varepsilon/2$, then condition on its output and run Theorem E.7 with $\eta = \varepsilon/2$. By the tower property,

$$\begin{aligned}\mathbb{E}[f(S)] &\geq (1-\eta)\mathbb{E}[F(x)] \\ &\geq (1-\eta)(1-1/e-\delta)\mathrm{OPT} \\ &\geq (1-1/e-\varepsilon)\mathrm{OPT}. \end{aligned} \tag{E.23}$$

The published phase uses $O(n\,\mathrm{poly}(1/\varepsilon, \log n))$ value and rank queries and supplies $t = O(1/\varepsilon)$ bases. Substitution into (E.18) gives

$$O\big(r\varepsilon^{-3}\log^2(2r)\big) \tag{E.24}$$

additional rank queries and no value queries. Since $r \leq n$, this term is absorbed into $O(n\,\mathrm{poly}(1/\varepsilon, \log n))$. □

For fixed accuracy, the query flow is therefore

$$\underbrace{\widetilde{O}(n)}_{\text{published fractional phase}} \quad + \quad \underbrace{\widetilde{O}(r)}_{\text{alternating-walk rounding}} \quad = \ \widetilde{O}(n). \tag{E.25}$$

The new idea changes only the second term.

F INDEPENDENCE ACCESS: GEOMETRIC REVERSE-WALK ROUNDING

F.1 REVERSING THE WALK CHANGES WHICH ENDPOINT HURTS

The rank-oracle rounder works because the unfavorable endpoint is the one we chose uniformly. An independence oracle lets us search the other way: fix an entrant and find an element to delete. This reverses the sign of the endpoint difference. Now the unfavorable endpoint is where the walk *ends*, and choosing its start uniformly does not control that end.

We repair the problem in two steps. First, stop at a geometric time, so the endpoint loss becomes p times the marginal charge of an element actually removed by the update. Second, charge all removals in a merge together. Once removed, a disagreement element never returns, and diminishing returns lets a deletion-only comparison path pay for every charge from one initial value budget. The expected walk length is $1/p$, while the loss of the whole merge is at most p.

The input remains a weighted base list and an allowed loss η; the output is one base. We keep the sequential merges of Algorithm E.2, replace its walk update by **GeometricUpdate**, and use $p = \eta/(t-1)$ in every merge. Call this procedure **IndependenceRound**. When $t = 1$, return that base, and when $r = 0$, return the empty base. We prove the following guarantee, with its full analysis below.

Theorem F.1 (Nearly linear independence-oracle rounder)**.** *Given* (A.1) *and* $0 < \eta < 1$*, the algorithm below returns a base* S *with*

$$\mathbb{E}[f(S)] \geq (1-\eta)F(x). \tag{F.1}$$

It uses no value queries and $O(r(t-1)^2\eta^{-1}\log(2r))$ *expected independence queries. If* $t = 1$*, the supplied base is returned immediately.*

For reference, the tangent inequality used here is

$$F(x + \gamma(\mathbf{1}_v - \mathbf{1}_u)) \geq F(x) + \gamma(\partial_v F(x) - \partial_u F(x)). \tag{F.2}$$

F.1.1 DELETION-SIDE FUNDAMENTAL-CIRCUIT SEARCH

For a base B and $e \notin B$, let $C_B(e)$ be the unique fundamental circuit contained in $B + e$; see, for example, Oxley (Oxley, 2011). If $Q \subseteq B$, then

$$(B \setminus Q) + e \in \mathcal{I} \quad \Longleftrightarrow \quad Q \cap \big(C_B(e) \setminus \{e\}\big) \neq \varnothing. \tag{F.3}$$

Indeed, $C_B(e)$ is the only circuit in $B + e$: the queried set is independent exactly when at least one of its elements has been deleted. Notice that the query in (F.3) contains at most r elements.

Lemma F.2 (Fundamental-circuit binary search: **DeleteSearch**). *Let $R \subseteq B$ be known to meet $C_B(e) \setminus \{e\}$. An element $b \in R$ such that $B - b + e$ is a base can be found using at most $\lceil \log_2(2|R|) \rceil$ independence queries. We write $b =$ **DeleteSearch**(B, e, R) for this published subroutine (Kobayashi and Terao, 2024, Lemma 2.1).*

Proof. Maintain a candidate set R' known to meet the circuit. Split it into two parts R_1, R_2 whose sizes differ by at most one and query $(B \setminus R_1) + e$. If the answer is independent, then (F.3) says that R_1 meets the circuit, so retain R_1. Otherwise R_1 misses the circuit, and the promise implies that R_2 meets it; retain R_2. The candidate set halves at every query. At a singleton, the remaining element is a valid deletion and the resulting independent set has size r, hence is a base. □

This published primitive is recorded in (Kobayashi and Terao, 2024, Lemma 2.1); it is not a new contribution of the walk rounder. This is the orientation available to an independence oracle: fix one entrant e, then group-test which current base elements may be deleted. Every query is a subset of $(B + e)$ of size at most r.

F.2 Why the fixed-length rank-oracle walk does not transfer

Try the simplest possible adaptation: use the deletion search above to build a reverse walk, but keep its length fixed. The next calculation shows exactly where the rank-oracle proof stops working.

A fixed reverse walk exposes the wrong endpoint. Consider the most direct fixed-length analogue. Reveal L links (v_j, u_j, v_{j+1}), choose J uniformly from $\{0, \dots, L-1\}$, and apply one of the two exchanges at link J with the branch probabilities $\beta/(\alpha+\beta)$ and $\alpha/(\alpha+\beta)$: replace u_J by v_J in B_1, or replace v_{J+1} by u_J in B_2. Here B_1, B_2 have coefficients α, β, and the walk guarantees both indicated exchanges are feasible. Let x be the current full fractional point. If $a_j = \partial_{v_j} F(x)$ and $c = \alpha\beta/(\alpha+\beta)$, the branch calculation used below gives

$$\begin{aligned}\mathbb{E}[F(x') \mid x, \text{walk}] &\geq F(x) + \frac{c}{L} \sum_{j=0}^{L-1} (a_j - a_{j+1}) \\ &= F(x) + \frac{c}{L}(a_0 - a_L). \end{aligned} \tag{F.4}$$

Monotonicity makes $a_0 \geq 0$, but the unfavorable term is now a_L. Nothing in the deletion-side search makes v_L uniform in the current disagreement set. Thus the $1/d$ endpoint estimate from the rank-oracle proof is unavailable.

One can always use the crude one-coordinate estimate

$$\beta a_L \leq F(x). \tag{F.5}$$

Indeed, $v_L \in B_2 \setminus B_1$, so $x_{v_L} \geq \beta$; for a nonnegative monotone multilinear extension, $x_e \partial_e F(x) \leq F(x)$. Since $c/\beta \leq 1$, (F.4) and (F.5) guarantee only a loss of at most $F(x)/L$ per update. A merge may contain r updates, so this direct accounting loses as much as $rF(x)/L$. To limit one merge to a factor-p loss, it requires $L = \Omega(r/p)$. Building L links costs $O(L \log(2r))$ independence queries per update, and hence

$$O(rL \log(2r)) = O\big(r^2 p^{-1} \log(2r)\big) \tag{F.6}$$

queries per merge. This is quadratic rather than nearly linear in the rank. The calculation is not a lower bound against every conceivable fixed-length algorithm; it pinpoints why the direct transplantation of the rank-oracle proof fails.

Geometric stopping repairs the endpoint mismatch. For K with $\mathbb{P}[K = k] = p(1-p)^k$, the same derivative differences obey

$$\mathbb{E}[a_K - a_{K+1}] = pa_0 - p\mathbb{E}[a_{K+1}]. \tag{F.7}$$

The unfavorable endpoint now carries the desired factor p and, crucially, v_{K+1} is precisely the element removed in the second branch. Instead of bounding its derivative separately in every update, we charge the loss to the realized removal. Section F.5 proves that all removal charges in one merge sum to at most its starting value. At the same time, $\mathbb{E}[K+1] = 1/p$, so the expected search length is only $1/p$, with no factor r hidden in the stopping parameter.

F.3 THE GEOMETRICALLY STOPPED REVERSE WALK

Suppose two bases B_1, B_2 with coefficients $\alpha, \beta > 0$ are being merged. Other bases and coefficients remain fixed. Put

$$A = B_1 \setminus B_2, \qquad C = B_2 \setminus B_1, \qquad d = |A| = |C|. \tag{F.8}$$

Assume $d > 0$ and fix $0 < p < 1$.

F.3.1 ONE UPDATE

The update receives the two bases and a stopping parameter p and returns the bases after one exchange. Fix the starting-vertex and search rules before drawing the stopping variable. These rules define an infinite sequence

$$v_0, u_0, v_1, u_1, v_2, u_2, \dots \tag{F.9}$$

independently of how much of it is exposed. The algorithm computes only its required finite prefix, in the frozen pre-update bases.

Algorithm F.1. GeometricUpdate$(B_1, B_2, \alpha, \beta, p)$

$A \leftarrow B_1 \setminus B_2$; $C \leftarrow B_2 \setminus B_1$. Choose $v_0 \in C$ by the fixed rule.
Independently draw $K \geq 0$ with $\mathbb{P}[K = k] = p(1-p)^k$.
for $j = 0, \dots, K$ **do**
  $u_j \leftarrow$ **DeleteSearch**(B_1, v_j, A).
  $v_{j+1} \leftarrow$ **DeleteSearch**(B_2, u_j, C).
end for
if an independent coin of success probability $\beta/(\alpha+\beta)$ succeeds **then**

$$B_1 \leftarrow B_1 - u_K + v_K. \tag{F.10}$$

else

$$B_2 \leftarrow B_2 - v_{K+1} + u_K. \tag{F.11}$$

end if
return (B_1, B_2).

The searches guarantee

$$B_1 - u_j + v_j \text{ and } B_2 - v_{j+1} + u_j \text{ are bases.} \tag{F.12}$$

The branch probabilities cancel the derivative of u_j. The geometric law

$$\mathbb{P}[K = k] = p(1-p)^k \tag{F.13}$$

then turns the remaining derivative difference into a factor-p charge on the shifted endpoint v_{K+1}. That endpoint is removed in the second branch, allowing the global removal budget to apply.

The stopping time is independent of the prefix-consistent sequence. This independence, not the physical order in which random bits and oracle answers are exposed, is the condition used in Section F.4.

F.3.2 FEASIBILITY, PROGRESS, AND QUERY COST

Given $v_j \in C$, apply Lemma F.2 to B_1 with entrant v_j and candidate deletions A. The fundamental circuit must meet A. Otherwise it would be contained in $(B_1 \cap B_2) + v_j \subseteq B_2$, contradicting independence of B_2. This produces $u_j \in A$ such that $B_1 - u_j + v_j$ is a base.

Now apply the same search to B_2 with entrant u_j and candidate deletions C. If its fundamental circuit missed C, it would be contained in $(B_1 \cap B_2) + u_j \subseteq B_1$, again a contradiction. The search therefore returns $v_{j+1} \in C$ and proves (F.12).

One revealed link (v_j, u_j, v_{j+1}) uses at most $2\lceil \log_2(2r) \rceil$ independence queries. Hence one update uses

$$2(K+1)\lceil \log_2(2r) \rceil \tag{F.14}$$

queries. Repeated walk vertices are harmless. In (F.10), the inserted v_K becomes common to both bases and the removed u_K becomes absent from both; the disagreement size falls from d to $d-1$. The same statement holds for (F.11).

Example F.3 (A terminal endpoint need not be uniform)**.** Use $U_{2,4}$ with $B_1 = \{a, b\}$, $B_2 = \{c, d\}$ and equal coefficients. Take modular weights $(3, 2, 10, 1)$ and start at $v_0 = d$. A legal fixed search rule produces links $(d, a, c), (c, a, c), (c, a, c), \ldots$. Thus $a_0 = 1$ and $a_j = 10$ for every $j \geq 1$. If K is geometric with parameter p, the expected endpoint difference is $-9p$, not a uniform average over $\{c, d\}$. Since the common coefficient is $1/4$, the exact expected value loss is $9p/4$. The expected removal cost is $\frac{1}{2} \cdot \frac{3}{2} + \frac{1}{2} \cdot 5 = 13/4$, so the bound by $p\mathbb{E}W$ is valid. At $p = 1/5$, the initial value eight becomes 7.55 in expectation. The sequence is fixed independently of K; choosing a different search rule after learning K would not justify the geometric identity.

F.4 Geometric telescoping

At the current update, write the full fractional point as

$$x = \alpha \mathbf{1}_{B_1} + \beta \mathbf{1}_{B_2} + z, \tag{F.15}$$

where $z \geq 0$ is the contribution of bases not currently being merged. All derivatives in this section are evaluated at this fixed x. Define

$$c = \frac{\alpha\beta}{\alpha + \beta}, \qquad a_j = \partial_{v_j} F(x). \tag{F.16}$$

The proof has three layers. First, for a fixed stopping index, the two possible exchanges cancel the derivative of the intermediate element u_k. Second, summing the remaining difference against the geometric law produces a factor-p shifted endpoint. Third, that shifted endpoint is identified with an element actually removed by the update.

Lemma F.4 (One-update removal charge)**.** *Let W be the coefficient-weighted derivative of the element actually removed by the update:*

$$W = \begin{cases} \alpha \partial_{u_K} F(x), & \text{if (F.10) is executed,} \\ \beta \partial_{v_{K+1}} F(x), & \text{if (F.11) is executed.} \end{cases} \tag{F.17}$$

Then

$$\mathbb{E}[F(x') \mid x] \geq F(x) - p\mathbb{E}[W \mid x]. \tag{F.18}$$

Proof. **Step 1: fix the stopping index and cancel the middle derivative.** Condition on $K = k$ and on the revealed walk prefix. In the first branch, B_1 loses u_k and gains v_k, so the fractional point moves from x to

$$x^{(1)} = x + \alpha(\mathbf{1}_{v_k} - \mathbf{1}_{u_k}). \tag{F.19}$$

Both endpoints of this segment lie in $[0, 1]^V$: before the exchange, u_k belongs to B_1 and v_k does not, and after the exchange the same coefficient α has simply moved between these coordinates. Therefore Lemma A.11 gives

$$F(x^{(1)}) \geq F(x) + \alpha\big(\partial_{v_k} F(x) - \partial_{u_k} F(x)\big). \tag{F.20}$$

Likewise, the second branch replaces v_{k+1} by u_k in B_2, producing

$$x^{(2)} = x + \beta(\mathbf{1}_{u_k} - \mathbf{1}_{v_{k+1}}) \tag{F.21}$$

and hence

$$F(x^{(2)}) \geq F(x) + \beta\big(\partial_{u_k} F(x) - \partial_{v_{k+1}} F(x)\big). \tag{F.22}$$

The first branch has probability $\beta/(\alpha + \beta)$ and the second has probability $\alpha/(\alpha + \beta)$. Thus the coefficient multiplying each parenthesized derivative difference is

$$\frac{\beta}{\alpha + \beta}\alpha = \frac{\alpha}{\alpha + \beta}\beta = c. \tag{F.23}$$

Taking the branch expectation in (F.20)–(F.22), the negative and positive occurrences of $\partial_{u_k} F(x)$ cancel exactly:

$$\begin{aligned}\mathbb{E}[F(x') \mid x, K = k, \text{prefix}] &\geq F(x) + c\big[\partial_{v_k} F - \partial_{u_k} F + \partial_{u_k} F - \partial_{v_{k+1}} F\big] \\ &= F(x) + c(a_k - a_{k+1}). \end{aligned} \tag{F.24}$$

Step 2: sum the endpoint difference against the geometric law. Because the prefix-consistent infinite sequence is fixed independently of K, we may first regard $a_0, a_1, \ldots$ as fixed and then average over K. Put $q = 1 - p$. Reindexing the second series by $j = k + 1$ gives

$$\begin{aligned}\mathbb{E}[a_K - a_{K+1} \mid x] &= \sum_{k \geq 0} pq^k a_k - \sum_{j \geq 1} pq^{j-1} a_j \\ &= pa_0 + \sum_{j \geq 1} \big(pq^j - pq^{j-1}\big) a_j \\ &= pa_0 - p \sum_{j \geq 1} pq^{j-1} a_j. \end{aligned} \tag{F.25}$$

The weights pq^{j-1}, $j \geq 1$, sum to one and are exactly the law of $K + 1$. Hence the last term is $p\mathbb{E}[a_{K+1} \mid x]$. Monotonicity of f implies every multilinear derivative is nonnegative, in particular $a_0 \geq 0$. Dropping this favorable term from (F.24) and (F.25) yields

$$\mathbb{E}[F(x') \mid x] \geq F(x) - cp\mathbb{E}[\partial_{v_{K+1}} F(x) \mid x]. \tag{F.26}$$

Step 3: identify the shifted endpoint with a realized removal. In the second branch, the element v_{K+1} is removed from B_2, whose coefficient is β. This branch is chosen with probability $\alpha/(\alpha + \beta)$. Since the branch choice is independent of K and the revealed walk, its contribution to the conditional expectation of W is therefore

$$\mathbb{E}[W\, \mathbf{1}_{\{\text{second branch}\}} \mid x] = \frac{\alpha}{\alpha + \beta}\, \beta\mathbb{E}[\partial_{v_{K+1}} F(x) \mid x] = c\mathbb{E}[\partial_{v_{K+1}} F(x) \mid x]. \tag{F.27}$$

The first branch contributes the other nonnegative part of W. Thus

$$c\mathbb{E}[\partial_{v_{K+1}} F(x) \mid x] \leq \mathbb{E}[W \mid x]. \tag{F.28}$$

Substituting (F.28) into (F.26) proves (F.18). □

Why the geometric law matters. The shift from the distribution of K to that of $K + 1$ changes the weight of every terminal derivative by the same factor $1 - p$. Their difference is therefore a factor-p removal charge. A fixed stopping length has no such discount identity.

F.5 The global removal budget

The one-update bound would be too weak if W were paid separately at every disagreement. The point of this section is that removals are a one-use resource: an element removed from the disagreement never returns to it. We compare the real trajectory, which contains both insertions and deletions, with an auxiliary trajectory that keeps only the deletions. Multilinearity then makes the auxiliary losses telescope exactly.

Example F.5 (A deletion-only trajectory bounds larger derivatives)**.** Let all four elements cover the same unit feature and start from $x^0 = (1/2, 1/2, 1/2, 1/2)$, so $F(x^0) = 15/16$. Consider the legal realized merges that delete a and insert c, then delete b and insert d, each with coefficient $1/2$. The first removal charge is $1/16$. Before the second removal c has coordinate one, so the second actual charge is zero. The deletion-only points are $(0, 1/2, 1/2, 1/2)$ and $(0, 0, 1/2, 1/2)$, with values $7/8$ and $3/4$. Their total decrease is $3/16$, which dominates the actual charge $1/16$. Ignoring insertions makes the remaining marginal derivatives larger, which is the direction needed for the proof.

Lemma F.6 (Removal-budget lemma)**.** *Fix a realized trajectory of one merge. Let x^0 be its initial global point, and let W_s be the removal cost in update s. Then*

$$\sum_s W_s \le F(x^0). \tag{F.29}$$

Proof. **Step 1: removed elements are distinct.** At any time during a merge, every element is in one of four states relative to the two current bases: common to both, present only in B_1, present only in B_2, or absent from both. In either branch of an update, the inserted element moves from a one-base state to the common state, while the removed element moves from a one-base state to the absent state. No other membership changes. Consequently the disagreement size decreases by exactly one.

Each update removes an element from the unique one of B_1, B_2 containing it, so that element becomes absent from both bases. The inserted element becomes common to both. Neither type can reenter the symmetric difference. Consequently, removed elements are distinct and none is reinserted during this merge.

Step 2: define a deletion-only comparison path. Let e_s be the element removed at update s, and let γ_s be the coefficient α or β of the base from which it is removed. Thus

$$W_s = \gamma_s \partial_{e_s} F(x^s), \tag{F.30}$$

where x^s is the actual point immediately before the update. Define the deletion-only comparison trajectory

$$y^0 = x^0, \qquad y^{s+1} = y^s - \gamma_s \mathbf{1}_{e_s}. \tag{F.31}$$

This auxiliary path performs every actual deletion, in the same order and with the same coefficient, but ignores every insertion. Because an element is never removed twice or reinserted after removal, all coordinates of y^s remain nonnegative. More explicitly, if i_h is the element inserted at update h with coefficient γ_h, then immediately before update s the actual point has the form

$$x^s = y^s + \sum_{h<s} \gamma_h \mathbf{1}_{i_h}. \tag{F.32}$$

Repeated insertions cannot occur here: after insertion an element is common and no longer belongs to the disagreement. In particular, $y^s \le x^s$ coordinatewise.

Step 3: compare marginal derivatives. For the multilinear extension of a submodular function, $\partial_e F$ is nonincreasing when any coordinate other than e is increased. This is the continuous diminishing-returns property. Therefore $y^s \le x^s$ implies

$$\partial_{e_s} F(x^s) \le \partial_{e_s} F(y^s). \tag{F.33}$$

There is no difficulty if the two points differ at coordinate e_s: because F is affine in each coordinate separately, $\partial_{e_s} F$ is independent of the e_s coordinate itself.

Step 4: charge one removal and telescope. The point y^{s+1} is obtained from y^s by decreasing coordinate e_s by exactly γ_s. Affinity in that coordinate gives the exact identity

$$\gamma_s \partial_{e_s} F(y^s) = F(y^s) - F(y^{s+1}). \tag{F.34}$$

Combining (F.30), (F.33), and (F.34), each realized removal satisfies

$$W_s = \gamma_s \partial_{e_s} F(x^s) \le F(y^s) - F(y^{s+1}). \tag{F.35}$$

Summing (F.35) over all updates cancels every intermediate auxiliary value:

$$\sum_s W_s \le F(y^0) - F(y^{\mathrm{end}}) \le F(x^0), \tag{F.36}$$

where the last inequality uses $y^0 = x^0$ and nonnegativity of f. □

Proposition F.7 (One geometric merge)**.** *One complete merge returns a base of coefficient $\alpha + \beta$ and satisfies*

$$\mathbb{E}[F(x^{\text{after}}) \mid x^{\text{before}}] \geq (1-p)F(x^{\text{before}}). \tag{F.37}$$

It uses at most

$$2rp^{-1}\lceil \log_2(2r) \rceil \tag{F.38}$$

expected independence queries.

Proof. Let $m = |B_1 \setminus B_2|$ at the start of the merge. Every update reduces this number by one, so the merge contains exactly $m \leq r$ updates. Let $\mathcal{H}_s$ denote the full history before update s. Applying Lemma F.4 conditionally on this history gives

$$\mathbb{E}[F(x^{s+1}) \mid \mathcal{H}_s] \geq F(x^s) - p\mathbb{E}[W_s \mid \mathcal{H}_s]. \tag{F.39}$$

Take total expectations and sum from $s = 0$ to $m - 1$. The left- and right-hand intermediate values telescope, while the tower property removes the inner conditioning:

$$\mathbb{E}[F(x^m)] \geq F(x^0) - p\mathbb{E}\left[\sum_{s=0}^{m-1} W_s\right]. \tag{F.40}$$

Lemma F.6 holds for every realized trajectory, not only in expectation, so the last expectation is at most $F(x^0)$. This proves (F.37).

For the query bound, one link uses at most $2\lceil \log_2(2r) \rceil$ independence queries. The number of links exposed in one update is $K + 1$, and

$$\mathbb{E}[K+1] = \sum_{k \geq 0} (k+1)p(1-p)^k = \frac{1}{p}. \tag{F.41}$$

Thus one update costs at most $2p^{-1}\lceil \log_2(2r) \rceil$ queries in expectation. Multiplying by $m \leq r$ proves (F.38). Once $m = 0$, the two bases are identical; replacing their two coefficients by the single coefficient $\alpha + \beta$ leaves the fractional point unchanged. □

F.6 Proof and accounting for the rounder

After merging the first j original bases, their representative has coefficient $A_j = \sum_{i=1}^{j} \beta_i$; the other bases retain their coefficients. Combining identical bases does not change the fractional point. At the end the sole coefficient is one, so the point is integral.

$$A_j = \sum_{i=1}^{j} \beta_i. \tag{F.42}$$

F.6.1 Composition of the value guarantee

Proof of Theorem F.1. If $t = 1$, then $x = \mathbf{1}_{B_1}$, so returning B_1 gives $f(B_1) = F(x)$ without an oracle query. Assume henceforth that $t > 1$ and use

$$p = \frac{\eta}{t-1} \tag{F.43}$$

in every merge. Let X_j be the random fractional point after j merges, with $X_0 = x$. The representative base entering a later merge is random, but Proposition F.7 is a statement conditional on the entire point at the start of that merge. Consequently

$$\mathbb{E}[F(X_j) \mid X_{j-1}] \geq (1-p)F(X_{j-1}). \tag{F.44}$$

Taking expectations and applying this recurrence $t - 1$ times yields

$$\mathbb{E}[F(X_{t-1})] \geq (1-p)^{t-1}F(x). \tag{F.45}$$

Because $X_{t-1} = \mathbf{1}_S$ and the multilinear extension agrees with f on incidence vectors, $F(X_{t-1}) = f(S)$. Bernoulli's inequality gives $(1-p)^{t-1} \geq 1 - (t-1)p$. Substituting $p = \eta/(t-1)$ therefore gives

$$\mathbb{E}[f(S)] \geq (1-(t-1)p)F(x) = (1-\eta)F(x), \tag{F.46}$$

which proves (F.1).

Query accounting.

The expected query count is the product of four transparent quantities:

Quantity	Bound
links revealed per update	$\mathbb{E}[K+1] = 1/p$
independence searches per link	2
queries per search	$\lceil \log_2(2r) \rceil$
updates per merge	at most r

Hence one merge costs at most the amount in (F.38). There are exactly $t-1$ merges, so substituting (F.43) gives

$$2r\frac{(t-1)^2}{\eta}\lceil \log_2(2r) \rceil \tag{F.47}$$

expected independence queries. All decisions use only the explicit bases, random bits, and answers from the independence oracle; no step evaluates f or F. The bound is in expectation because the geometric variable has unbounded support, although its mean is $1/p$.

□

F.6.2 Accuracy specialization

In the common rounding regime

$$t = O(1/\varepsilon), \qquad \eta = \Theta(\varepsilon), \tag{F.48}$$

the expected cost specializes to

$$O\big(r\varepsilon^{-3}\log(2r)\big)\,. \tag{F.49}$$

This statement concerns only the cost of rounding the supplied decomposition; it does not include or assume a particular fractional algorithm.

G Independence access: multilevel preprocessing

G.1 Make sampling cheap without repeatedly scanning the same elements

The geometric rounder completes the integral step once a short fractional base list is available. Finding that list still has the preprocessing bottleneck from Section C. We retain the familiar plan: select a prefix S, then apply continuous greedy to the residual objective $g(I) = f(S \cup I) - f(S)$ on M/S. What changes is how we find S.

The goal is to reduce the *sum of sample ranges along a feasible direction*. For a known scale $\mathrm{OPT} \le \Gamma \le 3\mathrm{OPT}$, we want

$$A_S := \max_{I \in \mathcal{I}(M/S)} \sum_{e \in I} f(e \mid S) \le \rho\Gamma, \qquad \rho = \mathrm{polylog}(n). \tag{G.1}$$

Section H will explain how this bound makes each marginal estimate cheap. The scale comes from standard threshold greedy; its specification and proof are at the end of this section.

Example G.1 (One choice can remove a hundred redundant ranges)**.** Suppose a rank-100 uniform matroid has 100 elements all covering the same unit-weight feature. Initially their singleton sum is 100 although $\mathrm{OPT} = 1$. Selecting any one of them makes all residual marginals zero. This is why selecting a prefix can help sampling so much. In a general matroid, however, a selected element can block a valuable part of an optimum. We must pay for that possibility while reducing the ranges.

Fix a threshold θ and call an element high when its marginal at the current prefix is at least θ. On a candidate independent set, we use a simple alternative: select a uniformly random high element if there are many, or identify almost all positions as low if there are few. Uniform selection makes the accepted gain pay for the expected value blocked in a comparison optimum. Low certificates remain valid as S grows, by diminishing returns.

The difficult part is finding new candidates cheaply. A scan of one base may produce many low certificates, yet rebuilding that base may require a scan of the whole ground set. Repeating this can be expensive. Several bases help only if their certificates concern *different elements*. Quanrud's packing algorithm provides exactly the structure we need: either the bases have large disjoint private subsets, or a substantial part of the rank lies in a small contraction minor. In the first case, each batch certifies enough distinct low elements to pay for rebuilding. In the second, we solve the same task inside the smaller minor. Repeating this idea at several levels, rather than using it just once, is what brings the exponent down to one.

G.2 The whole preprocessing algorithm

Precondition returns a feasible prefix S and the factor ρ in (G.1). The next section feeds these to continuous greedy on M/S, adds S to each returned residual base, and passes that short base list to the geometric rounder. The bases used *during preprocessing* are only temporary candidate sets for finding S; they are not the fractional solution that will be rounded.

How can we bound the sum in (G.1) without examining every feasible set? Fix a threshold θ. We only need to make the *rank of the high elements* small. Then no feasible direction can contain many marginals above θ, even if many dependent high elements remain. Repeating this task at decreasing thresholds controls the whole sum. Thus the outer loop is simple: finish the current threshold, lower θ, and repeat.

The work inside one threshold is organized around **ReduceRank**. One call reduces the current high rank by a prescribed fraction, or reaches the stopping floor. It uses **PackOrMinor** to decide where to work: process several bases with distinct private positions, or recurse on a smaller minor carrying much of the rank. **ProcessBases** handles the first branch. On each candidate base, **DensityCheck** either supplies a uniformly chosen high element to select, or certifies many low elements that can leave the current search. Figure 1 shows how these calls fit together; the following subsections explain them in this order.

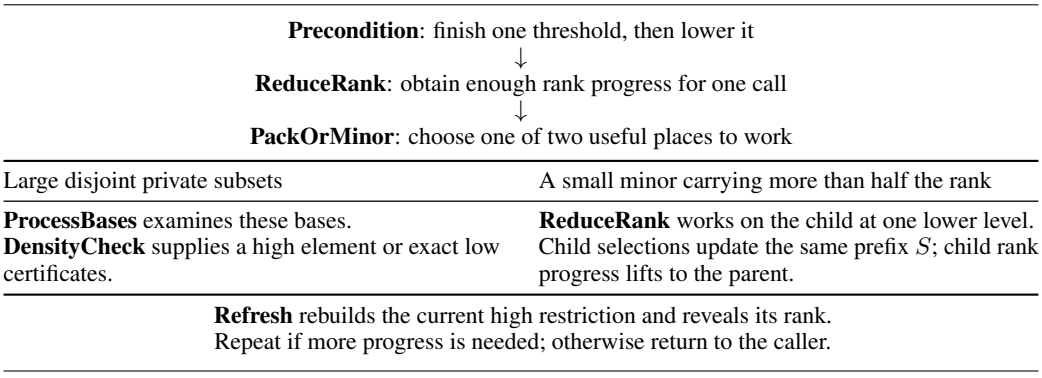


Figure 1: The calls within preprocessing. The left branch pays for rebuilding by certifying distinct low elements; the right branch obtains rank progress from a smaller problem. At level one, packing is skipped and only one base is processed.

There are two different kinds of change. Selecting a high element enlarges S permanently and contracts it from the feasible residual problem. Filtering a low element only removes it from the search at the current threshold: it may become relevant when the threshold drops. Consequently each new pass starts from the full residual matroid M/S. A recursive minor may also contract an auxiliary set to simulate feasibility; those elements are not selected into S.

The following is the outer algorithm. It receives M, f and a scale $\mathrm{OPT} \le \Gamma \le 3\mathrm{OPT}$, supplied by **EstimateScale** (Lemma G.16). The quantities H_ℓ are stopping ranks: below them the high rank is already small enough. Their exact values and the recursion depth L_* are specified below.

Algorithm G.1. Precondition$(M, f, \Gamma; \kappa, \zeta)$

$S \leftarrow \varnothing$; choose L_*, H_ℓ, ρ and the query cap as below.
for $\theta = \Gamma, (1-\kappa)\Gamma, \ldots,$ through the first $\theta \le \kappa\Gamma/r$ **do**
  $h_0 \leftarrow \lceil 4\Gamma/\theta \rceil$.
  $N \leftarrow$ **Refresh**$(M/S, S, \theta)$, starting from the full residual ground set.
  while $\mathrm{rank}(N) \ge H_{L_*}$ **do**
    $N \leftarrow$ **ReduceRank**(N, S, θ, h_0, L_*).
    if a subroutine reports an abort or the global cap is reached **then**
      return $(S, \rho, \mathrm{abort})$.
    end if
  end while
end for
return $(S, \rho, \mathrm{complete})$.

The rank is the size of the basis returned with each fresh state; it is not a rank-oracle query. Each new threshold restores the full residual matroid. Filtering within a pass does not delete elements from later passes.

After one pass, a feasible direction contains only a bounded number of marginals above its threshold. Across the passes, its ith largest marginal is bounded by roughly $\Gamma\,\mathrm{polylog}(n)/i$. Summing this harmonic sequence gives (G.1). Selection must also preserve enough value for the later optimization. That is the purpose of the uniform choice inside **DensityCheck**; the proof tracks its payment separately from the work saved by filtering.

G.3 ReduceRank: finish one piece of a threshold pass

The outer loop asks for a small final high rank, but an individual **ReduceRank** call has a more manageable job: reduce its input rank d_0 to $(1-\Delta_\ell)d_0$, or below the floor H_ℓ. Its input is a *fresh* search state N: all active elements are high at the current S, and a supplied basis reveals its rank. Its output is another fresh state, with any selections already added to the global prefix. The caller can therefore check the new rank and either invoke it again or finish the threshold pass.

At level one, the procedure repeatedly processes a single basis and rebuilds it. At higher levels, **PackOrMinor** gives one of two ways to improve this expensive pattern (Lemma G.3):

- If every supplied base has a large private subset, process the bases as a batch. A completed batch identifies many *different* low elements, so it makes substantial progress before the next full scan.
- Otherwise, a small contraction minor contains more than half the current rank. Run the same rank-reduction task there at one lower level. Its rank decrease is enough to finish this parent call after a refresh.

Filtering need not lower rank immediately: many redundant low elements may leave before the rank changes. The first branch therefore repeats batches until the rank target is met. The second branch needs only one child call.

Algorithm G.2. ReduceRank$(N, S, \theta, h_0, \ell)$

Let d_0 be the rank of the input's supplied basis.
if $d_0 < H_\ell$ **then**
  return N.
end if
$q \leftarrow \max\{2, |E(N)|/d_0\}$; $d_{\mathrm{goal}} \leftarrow (1 - \Delta_\ell)d_0$.
while the current rank $k > d_{\mathrm{goal}}$ and $k \geq H_\ell$ **do**
  if $\ell = 1$ **then**
    Process one current basis using **ProcessBases**; refresh N.
    continue
  end if
  $b \leftarrow b_\ell(q)$; obtain $(B_1, \ldots, B_b, P_1, \ldots, P_b, d, N')$ from **PackOrMinor**.
  if $d \leq k/2$ **then**
    Process the b bases using **ProcessBases**; refresh N.
  else
    Save $S_{\mathrm{old}} \leftarrow S$.
    $N' \leftarrow$ **ReduceRank**$(N', S, \theta, h_0, \ell - 1)$.
    Contract $S \setminus S_{\mathrm{old}}$ in the parent, then refresh the parent at the updated S.
    return the refreshed parent.
  end if
end while
return N, refreshing once if a batch reached a stopping rank by contractions.

All batches receive $d_{\mathrm{goal}}, H_\ell$. Detected aborts propagate immediately.

A recursive call changes the same selected prefix S, so the parent first contracts those newly selected elements and then scans at the updated marginals. It does not have to replay the child's filtering decisions: every element the child certified low is still low and is removed by the parent's refresh. This is how the child's work becomes progress for the original threshold pass.

Example G.2 (A child call finishes its parent)**.** At level two, consider a parent of rank $20a$ whose small minor has rank $12a$, with a large enough that the stopping floors do not intervene. The contracted part outside the child has rank $8a$. A level-one call halves the child's rank to at most $6a$. After the parent refreshes, its rank is at most $8a + 6a = 14a$, already below the level-two target $15a$. The child need not solve the entire threshold problem: its partial progress is enough for the parent to return. The auxiliary contracted part contributes to this rank calculation, but is not added to the selected prefix.

For completeness, the algorithm uses the following parameters. Their balance is explained in Section G.6; the stopping floors ensure that a batch has many more positions than the density-test tolerance.

Fix a threshold. For integers $\ell \geq 1$, put

$$p_\ell = \frac{2^\ell}{2^\ell - 1}, \qquad \Delta_\ell = 2^{-\ell}, \qquad H_\ell = C4^\ell h_0, \tag{G.2}$$

where C is a sufficiently large absolute constant.

For $\ell \geq 2$, use

$$b_\ell(q) = \left\lceil q^{2/(p_{\ell-1}+1)} \right\rceil. \tag{G.3}$$

The stopping floors ensure $k \geq 16h_0$ whenever the batch-counting lemma is needed, by taking the absolute constant C sufficiently large.

G.4 PackOrMinor: find distinct work or a smaller problem

ReduceRank needs to decide whether scanning several bases will save work. Merely asking for several bases does not answer this: they could all expose the same low elements. **PackOrMinor**

receives the current matroid and a desired number b of bases. It returns bases together with private subsets, or identifies a small minor on which rank reduction is cheaper. Thus its output tells the caller *which branch to take*; it does not select objective elements itself.

The bases need not be disjoint. Large disjoint subsets are enough to avoid counting the same filtering work repeatedly. If those subsets are too small, the certificate supplies the alternative minor. The next lemma derives exactly this guarantee from Quanrud's published packing result; the packing algorithm itself is used as a black box.

Lemma G.3 (Private subsets or a small obstruction minor)**.** *Given $N = (X, \mathcal{J})$ of rank m and $b \geq 2$,* **PackOrMinor**(N, b) *uses $\widetilde{O}(|X| + bm)$ independence queries to return bases $B_1, \ldots, B_b$, pairwise disjoint private sets $P_i \subseteq B_i$, and $0 \leq d \leq m$ with $|P_i| \geq m - d$. Either $d = 0$ and the bases are disjoint, or a set U satisfies*

$$|U| \leq 2bd, \qquad m - r_N(X \setminus U) = d.$$

There is then a rank-d contraction minor on at most $2bd$ elements, with every query simulated by one original independence query.

The two useful outcomes. If $d \leq m/2$, each private set has at least $m/2$ elements, so scans can be charged to different low elements. If $d > m/2$, the child retains more than half the rank but has at most $2b$ elements per unit of rank. We solve the same problem more cheaply in that child.

Proof. We use the following form of Quanrud's result (Quanrud, 2024, Section 3.1, Theorem 3.3, proof of Lemma 4.1, and Theorem 4.2). At constant packing accuracy, on a rank-m matroid $N = (X, \mathcal{J})$ and parameter $b \geq 2$, $\widetilde{O}(|X| + bm)$ independence queries produce either b disjoint bases or maintained bases and levels with some $j \geq 1$ such that

$$|X_{\leq j}| \leq 2b\big(m - r_N(X_{>j})\big). \tag{G.4}$$

The level invariants guarantee that each $B_{i,\geq j}$ spans $X_{>j}$, and every element occurring in two bases has level zero. These are the specific published facts used; the push–relabel algorithm is not reproved. Rank quantities in its certificate are properties, not requested oracle answers.

In the disjoint case take $P_i = B_i$ and $d = 0$. Otherwise take the level j in (G.4), let $U = X_{\leq j}$, $d = m - r_N(X_{>j})$, and $P_i = B_{i,\geq j}$. Spanning implies $|P_i| \geq r_N(X_{>j}) = m - d$. A shared private element would occur in two bases and have level at least one, contradicting the level-zero property. Thus the private sets are disjoint and the displayed certificate follows.

Construct a basis J of $X \setminus U$ greedily, and find $F_0 = \mathrm{cl}_N(J)$ by testing $J + e$ for all e. These scans cost $O(|X|)$ independence queries and also reveal $d = m - |J|$ without a rank oracle. The matroid $(N/J)|_{X \setminus F_0}$ has rank d, and its ground set is contained in U, hence has size at most $2bd$. Adjoin the known independent set J to every query in the minor. The output explicitly includes J and the child ground set, so the recursive caller can execute every child query. This proves the guarantee. □

Example G.4 (Private sets and a sparse obstruction)**.** Take the direct sum of one coloop s and $U_{2,6}$, with three disjoint pairs in the latter component. The three bases all contain s, but their pairs are disjoint private sets. Here $m = 3$, $d = 1$, and each private set has size two. Shared bases can therefore still provide disjoint filtering work. For a sparse example, take three coloops and $U_{2,20}$. The rank is five, and contracting the rank-two component exposes a rank-three minor on the three coloops. This illustrates the structural certificate; whether a threshold procedure needs to recurse also depends on its objective threshold and its stopping floor.

G.5 ProcessBases: turn private positions into useful progress

The packing branch has found good candidate bases. **ProcessBases** must now extract useful work from them before paying for another scan of the whole ground set. It receives those bases and the caller's stopping ranks, updates the selected prefix if it finds suitable high elements, and returns a reduced search state for the caller to refresh.

Consider just one of its bases. If many positions are still high, choosing a uniform high element gains value while controlling what the selection can block. If few are high, scanning the whole base is worthwhile because most positions can be certified low. **DensityCheck** makes exactly this

choice. It receives the candidate independent set A, the current S, threshold θ, count target h_0, and failure allowance α. The return value tells the batch to accept one element, record low certificates, or propagate a detected abort. We use its guarantee here and defer its sampling implementation to Section G.9.

Lemma G.5 (Density-test guarantee)**.** *Put $m = |A|$. For $h_0 \geq 1$ and $0 < \alpha < 1$, except with probability at most α, the call does not abort, an accepting call has at least h_0 high candidates, and a filtering call leaves fewer than $2h_0$ high candidates. Conditional on the fixed input, pilot outcome, and successful search, acceptance is exactly uniform among all high elements. An accepting call costs $O((m/h_0)\log(4/\alpha))$ value queries; filtering adds $O(m)$.*

Uniformity has a specific purpose. Match the h high candidates to distinct removable positions of an analytical base containing S. The total removal-marginal mass is at most OPT, so a uniform candidate causes expected comparison loss at most OPT/h. Its accepted gain is at least θ. Lemma G.14 proves this claim, including the case where the candidate set and comparison base overlap.

For the threshold pass choose

$$h_0 = \lceil 4\Gamma/\theta \rceil. \tag{G.5}$$

A sound acceptance gains at least θ and loses at most $\mathrm{OPT}/h_0 \leq \theta/4$ from the comparison base in expectation. Furthermore, over the entire pass there are at most $\Gamma/\theta \leq h_0/4$ acceptances: their gains telescope to at most $f(S) \leq \mathrm{OPT} \leq \Gamma$. This bound holds even if a pilot is unsound, since every accepted marginal is checked exactly.

There are consequently few acceptances in one threshold pass. Most of the work must be justified by filtering, and this is where distinct private elements matter.

Keep the other candidate bases usable. One selection changes the residual matroid, so the batch has to repair every unfinished base before continuing. Rebuilding them all from the ground set would lose the saving. Instead, one deletion from each base suffices after a selected element is contracted. Low certificates are handled differently: the batch records them and postpones their deletion until it returns. Otherwise filtering one base could invalidate the next base before its turn. **Refresh** then performs a single scan to remove the lows and compute the new rank.

A state N is a restriction of a contraction of M/S. The selected prefix S is global. A known auxiliary independent set J may be adjoined to feasibility queries, but its elements are not selected for the objective: all marginal queries remain $f(e \mid S)$, not $f(e \mid S \cup J)$. An independent set in the child is independent in M/S, so its acceptances are globally feasible and the uniform-exchange lemma still applies.

We use two elementary state operations. The first returns a current high restriction; the second preserves a supplied basis after an acceptance.

Lemma G.6 (Refresh and one-element repair)**. Refresh** *scans the active ground set of N, deletes its currently low elements, and greedily computes a basis. It returns this fresh restriction, a basis, and its rank using $O(|E(N)|)$ value and independence queries. If a nonloop e is accepted, an old basis B is converted to a basis of N/e by deleting e when $e \in B$, and otherwise by deleting $u =$ **DeleteSearch**(B, e, B) in $O(\log(2n))$ independence queries. All repair queries are made before contracting e.*

Proof. Each value check uses the current global prefix. Greedy adds an element exactly when independence is preserved; matroid augmentation makes the resulting maximal independent set a basis of the restriction, so its size is the rank. Each query in a minor is simulated by adjoining its known contracted independent sets. This proves the first bound without rank access. For repair, if $e \in B$, then $B - e$ is a basis of N/e. Otherwise Lemma F.2, the primitive of (Kobayashi and Terao, 2024, Lemma 2.1), finds u with $B - u + e$ a basis of the old state. Hence $B - u$ is a basis of N/e. Repairing every unfinished basis in the old state and then committing the contraction makes these conclusions hold simultaneously. □

With these maintenance rules, the batch repeatedly tests its current base until that base has been filtered or the caller's rank target is met. The shared prefix S changes only on an acceptance. The returned state contains all contractions and recorded low-element deletions; the caller's refresh supplies a new basis and exact rank.

Algorithm G.3. ProcessBases$(N, S, \theta, h_0; B_1, \dots, B_b, d_{\mathrm{goal}}, H)$

$D \leftarrow \varnothing$.
for $i = 1, \dots, b$ **do**
  while basis B_i has not been filtered **do**
    $R \leftarrow$ **DensityCheck**$(B_i, S, \theta; h_0, \alpha)$.
    if $R = \mathrm{abort}$ **then**
      return abort.
    end if
    if $R = \mathrm{Filter}(D_i)$ **then**
      $D \leftarrow D \cup D_i$; mark B_i filtered.
    else
      Let $R = \mathrm{Accept}(e)$. Repair every unfinished basis using Lemma G.6.
      $N \leftarrow N/e$; $S \leftarrow S + e$.
      if $\mathrm{rank}(N) \le d_{\mathrm{goal}}$ or $\mathrm{rank}(N) < H$ **then**
        return the current state restricted by deleting D.
      end if
    end if
  end while
end for
return the current state restricted by deleting D.

The caller refreshes the returned state. Until the final restriction, rank decreases by one per contraction and is known without a rank query.

The reason a batch pays for itself is that at most $h_0/4$ acceptances occur during the entire threshold pass. Each deletes at most one private position from any unfinished basis. A later filtering test leaves fewer than $2h_0$ high positions, so a large private set must contribute many low elements. The private sets are disjoint, allowing those contributions to be added across bases.

Example G.7 (Count private positions, not repeated appearances)**.** Suppose $k = 1600$, $h_0 = 100$, and four maintained bases each have a private set of 800 elements. At most 25 acceptances occur in the whole threshold pass. When a basis is filtered, fewer than 200 of its remaining positions are high. Its original private set therefore supplies at least $800 - 25 - 200 = 575$ low elements. The four contributions concern different elements, so they give at least 2300 distinct certificates. Without private sets, four scans could certify the same 575 elements four times. This is the saving in the rebuilding cost; the bases still need repair after acceptances.

Lemma G.8 (A completed batch removes distinct elements)**.** *Suppose the input rank is $k \ge 16h_0$, the b supplied bases have pairwise disjoint private sets of size at least $k/2$, and all density tests are sound. Unless the batch stops early by a rank target, it certifies at least $bk/4$ distinct low elements. Its value and independence cost is $\widetilde{O}(bk)$, excluding the caller's final refresh.*

Proof. Each private set loses at most $h_0/4$ positions to contractions before its basis is filtered. Fewer than $2h_0$ remaining basis positions are high. It therefore contributes at least $k/2 - h_0/4 - 2h_0 \ge k/4$ low positions. These sets are disjoint, and low marginals remain low after subsequent acceptances. Deferring their actual deletion preserves the other maintained bases while retaining all these valid certificates. There is at most one terminal full scan per basis, costing $O(bk)$. Every pilot or search costs $O((k/h_0)\log(4/\alpha))$ and is followed by an acceptance or the basis's final scan. The at most $h_0/4$ accepting calls cost $\widetilde{O}(k)$ in total; the b terminal pilots cost $\widetilde{O}(bk)$. Repairs cost $O(bh_0 \log(2n)) \le \widetilde{O}(bk)$. This counts every operation within the batch. The case of one basis also applies, taking that entire basis as its private set. □

G.6 Why the recursive work becomes nearly linear

We now have the complete chain of operations: density tests supply selections or low certificates; a batch turns private positions into distinct certificates; and a rank-reduction call chooses between such

batches and one smaller child. The remaining question is how much work can occur before the outer threshold loop finishes.

Why the child runs one level lower. The sparse branch relies on *one child call finishing its parent*. This is why the level decreases. A level-ℓ parent wants to remove a $\Delta_\ell = 2^{-\ell}$ fraction of its input rank. The child contains more than half the current rank, but generally not all of it. Removing the same fraction of the child's rank can therefore be insufficient.

For example, revisit the rank-$20a$ parent and rank-$12a$ child from Example G.2, with all ranks above the stopping floors. The level-two parent must reach rank $15a$. A *level-two* child call is allowed to stop at rank $9a$: it has removed its required quarter. Lifting this result gives only the parent bound $8a + 9a = 17a$, which misses the parent's target. A *level-one* child call instead reaches rank $6a$, giving the sufficient parent bound $8a + 6a = 14a$. The same-level call might do better, but its guarantee does not require it.

In general, if the current parent rank is k and the child rank is $e > k/2$, a same-level call guarantees only a loss $\Delta_\ell e$ in the rank-lifting bound. The lower-level call guarantees

$$\Delta_{\ell-1} e = 2\Delta_\ell e > \Delta_\ell k.$$

This is enough even when the parent has already made some progress from its original rank. If the child stops at its floor instead, the relation $H_\ell = 4H_{\ell-1}$ gives the same conclusion, as the proof below shows. Thus a lower level asks for a larger fractional rank decrease; the smaller minor makes that stronger request affordable.

With a same-level child, the parent would have to allow further calls or change its stopping rule. Those calls could in turn require further same-level calls. The single-child recurrence and the bound of at most $\ell - 1$ recursive descents would no longer follow. Using level $\ell - 1$ gives both properties, which are the call-count controls used in Lemma G.10. This explains the choice in our algorithm; the example is a failure of the same-level *guarantee*, not a lower bound against every possible same-level design.

Balance the work in the two branches. One packing level improves the cost, but does not yet make it nearly linear. To see why recursion helps, let the current rank be d, and let q denote roughly the number of elements per unit of rank. A direct rebuild-after-filtering procedure can cost $\widetilde{O}(q^2 d)$.

Packing b bases changes the filtering cost to roughly $(q^2/b)d$, because each completed batch removes b times as many distinct elements. The other outcome is a child with aspect ratio $O(b)$ and a constant fraction of the parent's rank. If the previous-level procedure costs $\widetilde{O}(q^p d)$, that child costs roughly $b^p d$. Choosing b to balance q^2/b and b^p improves the exponent from p to $2p/(p+1)$. The full proof below tracks the rank targets, constant factors, and logarithms suppressed in this explanation.

Example G.9 (A second level, then a third)**.** At the base level $p = 2$. For $q = 64$, choose $b = q^{2/3} = 16$. Both q^2/b and b^2 equal 256, compared with $q^2 = 4096$. Thus the second level has exponent $4/3$. Applying the same balance again gives $8/7$, then $16/15$, and so on. A depth of order $\log \log n$ leaves only logarithmic overheads and a bounded extra factor above linear. These counts illustrate the cost balance, for ranks above the stopping floors; they are not measured running times.

Lemma G.10 (One rank-reduction epoch)**.** *On the simultaneous sound-test event, a fresh rank-d minor on at most qd elements, $q \geq 2$, is processed by Algorithm G.2, which returns a fresh residual minor of rank below H_ℓ or at most $(1 - \Delta_\ell)d$. It uses*

$$K_0^\ell \, q^{p_\ell} d \, \mathrm{polylog}(n/\zeta)$$

value and independence queries for an absolute K_0, with the degree of the polylogarithm independent of ℓ. Every acceptance satisfies the comparison payment of Lemma G.14.

Proof. **Base level.** Build a basis by a greedy scan, then repeatedly apply the density test on that basis. A sound dense test supplies a uniformly selected high element; contract it and continue on the remaining basis. A failed density test filters all but fewer than $2h_0$ positions. Rebuild after that failure. While rank remains at least $d/2$ and H_1, a failed basis has $\Omega(d)$ low elements, since $H_1 = 4Ch_0$. Low elements are removed once, so there are $O(q)$ failures among at most qd elements.

Each rebuilding scan costs $O(qd)$. Across all successes there are at most $h_0/4$ acceptances, costing $\widetilde{O}(d)$ value probes on any one basis and no larger total than the stated $\widetilde{O}(q^2 d)$ budget. Rank and freshness are known after a rebuilding scan. Thus the base level costs $\widetilde{O}(q^2 d)$ and has the required $1/2$ progress, since $p_1 = 2$.

Inductive algorithm. Let $p = p_{\ell-1}$ and choose $c = \lceil q^{2/(p+1)} \rceil$. While the current rank k exceeds the target $(1 - \Delta_\ell)d$ and the floor H_ℓ, rescan and apply Lemma G.3 with packing parameter c. Throughout a nonterminal epoch, $d/2 \le k \le d$, so original and current aspect ratios differ by at most a constant.

Dense case. If the obstruction rank is at most $k/2$, each of the c private sets has size at least $k/2$. Process the maintained bases as one batch, using the repair rule above after every acceptance. There are at most $h_0/4$ acceptances globally at this threshold. Each acceptance removes at most one private position from each unused base. Thus, as $k \ge C4^\ell h_0$, each later base still has a constant fraction of its original private positions. A failed density test leaves fewer than $2h_0$ high positions in the whole base; consequently it certifies $\Omega(d)$ of that base's remaining private positions as low.

Unless the epoch terminates early by rank progress, the completed batch therefore filters $\Omega(cd)$ distinct elements. Disjointness of the private sets is essential to count them once. At most $O(1 + q/c)$ dense batches can occur. Packing and fresh scans cost $\widetilde{O}(qd + cd)$ per batch; all c terminal base scans cost $O(cd)$. Pilot searches cost at most $\widetilde{O}(d)$ over all acceptances of this epoch, and circuit repairs cost $\widetilde{O}(ch_0) \le \widetilde{O}(cd)$. The dense work is therefore

$$\widetilde{O}\big((q + q^2/c)d\big), \tag{G.6}$$

using $c \le 2q$. The algorithm stops as soon as its target rank or floor is reached.

Sparse case and lifting. If the obstruction rank is $e > k/2$, the certificate gives a child minor of rank e on at most $2ce$ elements. It is formed by contracting a known basis J of a flat F_0 of rank $k - e$ and restricting outside F_0. Since $k \ge H_\ell = 4H_{\ell-1}$, we have $e > 2H_{\ell-1}$. Invoke **ReduceRank** once on this child at level $\ell - 1$. Let A be its accepted set and let e' be its returned residual rank.

Here is the rank-lifting inequality in detail. The child accepts a set independent with J, so contracting A leaves J independent of rank $k - e$. Let Z be the remaining child ground set. In the parent after contracting A, adjoining the flat F_0 contributes at most $k - e$ rank, and the rank of Z after also contracting J is e'. Hence the union has rank at most $(k - e) + e'$. A fresh parent scan may discard more elements, so its residual rank is no larger. Child-certified low elements remain low at the enlarged global S, and elements outside the child belong to the contracted flat, so this accounting covers the entire parent ground set.

If $e' < H_{\ell-1}$, the resulting rank is below $k/2 + H_{\ell-1} \le 3k/4$. Otherwise it is at most $k - \Delta_{\ell-1}e < (1 - \Delta_\ell)k$. Either outcome supplies the parent epoch's required progress. Thus a parent calls at most one sparse child; no family of parked children accumulates. The child work is bounded by its inductive cost, namely a constant times $K_0^{\ell-1} c^p d$ times the same fixed polylogarithm.

Balancing and depth factors. The two important terms are q^2/c and c^p. The choice of c balances them, up to constants from rounding, at $q^{2p/(p+1)}$. Thus $p_\ell = 2p_{\ell-1}/(p_{\ell-1} + 1)$, whose solution with $p_1 = 2$ is (G.2). Factors of two in child aspect ratios and ceilings contribute only an absolute multiplier, since $1 < p \le 2$. Choose K_0 large enough to absorb it and the dense contribution. Packing and density searches at each level incur fixed-degree logarithms; they are additive to the single recursive child cost. They are not multiplied as a new logarithmic factor at each depth. Induction therefore leaves K_0^ℓ times one fixed-degree polylogarithm, as stated. Feasibility and the uniform-comparison payment hold at every accepted step by the recursive-state conventions and Lemma G.14. □

Theorem G.11 (One threshold)**.** *On a residual ground set of at most N elements, repeated epochs at depth $L_* = \lceil \log_2(2 + \log(2N)) \rceil$ reduce fresh threshold rank below $O((\Gamma/\theta)\log^2(2N))$ using $N \operatorname{polylog}(n/\zeta)$ queries, on the sound-test event.*

Proof. Let $s = 2^{L_*} = \Theta(\log(2N))$. At rank d, use $q = \max\{2, N/d\}$ in Lemma G.10. Its main factor obeys

$$q^{p_{L_*}} d \le O\big(N \max\{2, N/d\}^{1/(s-1)}\big) = O(N).$$

The depth multiplier $K_0^{L_*}$ is also polylogarithmic in N. Every nonterminal epoch decreases rank by a $1/s$ fraction, so $O(s \log(2r))$ epochs suffice. Finally $H_{L_*} = C4^{L_*} h_0 = O((\Gamma/\theta)\log^2(2N))$. All logarithmic degrees remain absolute constants. This also handles $N/d < 2$ by the displayed maximum; no sparse-aspect special assumption is needed. □

G.7 From threshold ranks to cheap residual sampling

The recursive analysis proves that each pass is fast and leaves a small high rank. We now translate those rank certificates into the two facts the next section needs: small sample ranges on every feasible residual set, and enough retained value despite the commitment to S.

Filtering is temporary throughout this argument. The next, lower threshold starts again from the full residual matroid; low elements remain available to the eventual continuous-greedy phase. On the other hand, certificates from completed higher thresholds remain valid: later contractions cannot increase rank, and growing S can only decrease marginals.

Theorem G.12 gives both required conclusions: (G.1), and an analytical base $C \supseteq S$ satisfying

$$\mathbb{E} f(S) \ge (\mathrm{e}-1)\mathbb{E}[\mathrm{OPT} - f(C)] - K_{\mathrm{cmp}}\zeta \mathrm{OPT}. \tag{G.7}$$

The algorithm does not compute C. The first bound controls query cost; the second controls value lost by committing to S.

Theorem G.12 (Comparator-safe singleton envelope)**.** *For target failure ζ, Algorithm G.1 costs $O(n\,\mathrm{poly}(1/\kappa, \log(2n), \log(1/\zeta)))$ queries on its sound event and returns an independent S with*

$$A_S \le C_1 \Gamma \log^2(2n)\log(2r)/(1-\kappa).$$

Its acceptances can be coupled to a base $C \supseteq S$, initially an optimum base, with an absolute constant K_{cmp} such that

$$\mathbb{E}[f(S)] \ge (\mathrm{e}-1)\mathbb{E}[\mathrm{OPT} - f(C)] - K_{\mathrm{cmp}}\zeta\mathrm{OPT}.$$

The implementation can be capped at the same polynomial budget, losing only $O(\zeta\mathrm{OPT})$ in the eventual expected-value guarantee.

Proof. There are $O(\kappa^{-1}\log(r/\kappa))$ thresholds. Sum the preceding query bound and allocate the conditional density-test failures as in Section G.8. For the envelope, fix a residual independent set and order its marginal weights $w_1 \ge w_2 \ge \cdots$. For a weight above the final grid floor, choose the largest grid threshold $\theta \le w_i$. Its high restriction has rank at least i, because the first i elements are independent and have marginal at least θ. Its rank certificate and $w_i < \theta/(1-\kappa)$ imply

$$i w_i \le C_2 \Gamma \log^2(2n)/(1-\kappa).$$

Sum over $i \le r$ to get the harmonic factor. The smaller weights sum to at most $\kappa\Gamma$, absorbed by the bound.

For comparison, maintain C only analytically. Each sound acceptance has expected gain at least $\mathrm{e}-1$ times its expected exchange loss. The pilot and the independent uniform choice have the conditioning specified above, so the comparison lemma applies at each current history. Low filtering never changes C. Sum conditional expectations; exchange losses telescope to $\mathrm{OPT} - f(C)$, and gains telescope to $f(S)$. The expected number of unsound acceptances is $O(\zeta)$, each costing at most OPT. This gives the stated additive error. A budget abort has probability $O(\zeta)$; return a greedy base extending the current feasible prefix on that event. Such an event changes any final approximation comparison by at most $O(\zeta\mathrm{OPT})$, because integral feasible objective values lie in $[0, \mathrm{OPT}]$. This proves the capped formulation used next. □

Why our range factor differs from earlier preprocessing bounds. The factor ρ measures how large the residual singleton sum can be relative to the chosen value scale; it is not the approximation ratio. For a fixed scale, a smaller factor is a stronger guarantee. Here $\Lambda = \Gamma$ in the subsequent continuous phase, and $\mathrm{OPT} \le \Gamma \le 3\mathrm{OPT}$ refers to the *original* problem. Thus factors stated against Γ and against OPT can be compared up to an absolute constant.

Our logarithms come from two specific steps. First, the stopping floor is $H_\ell = C4^\ell h_0$. The depth $L_* = \Theta(\log\log n)$ that makes preprocessing nearly linear also makes $H_{L_*} = O((\Gamma/\theta)\log^2(2n))$ at threshold θ. Second, a bound on high rank controls the ith largest weight by $O(\Gamma \log^2(2n)/((1-\kappa)i))$; summing these weights incurs the harmonic factor $\log(2r)$. We therefore obtain $\rho = O(\log^2(2n)\log(2r)/(1-\kappa))$. The recursion permits a larger remaining high rank in exchange for cheaper processing, and the passage from ranks to a weight sum incurs another logarithm. These are losses in this construction and analysis, not lower bounds on what every independence-oracle preconditioner must lose.

The comparison below concerns successful preprocessing outputs and a scale within a constant factor of the original optimum. In its last row, $\kappa \le 1/2$, as in our final parameter choice.

Preprocessing method	Singleton-sum factor	Source and cost distinction
Ene–Nguyen	$O(1/\varepsilon)$	(Ene and Nguyen, 2019, full version, Lemma 13); maintains a dynamic maximum-weight base.
Buchbinder–Feldman–Schwartz	$O(r/(\lambda\varepsilon))$	(Buchbinder et al., 2017, Section 3.3, Algorithm 1); $1 \le \lambda \le r$ trades more preprocessing for fewer later samples.
This paper	$O(\log^2(2n)\log(2r))$	Theorem G.12; nearly linear preprocessing using value and independence queries.

In the notation of the cited BFS preprint, the chosen factor is $240r/(\lambda\varepsilon)+2$ (its rank is denoted by k). For fixed accuracy, the Ene–Nguyen factor is independent of n and gives a stronger asymptotic range bound than our displayed one. In the independence-oracle tradeoff, the balanced choice $\lambda = \Theta(\sqrt{r})$ instead leaves a factor $O(\sqrt{r}/\varepsilon)$; our factor is smaller when rank is sufficiently large. Taking $\lambda = r$ restores an $O(1/\varepsilon)$ factor, but costs $\widetilde{O}_\varepsilon(nr)$ preprocessing independence queries; see (Kobayashi and Terao, 2024, Section 3, Theorem 3.1 and preceding discussion). There is no uniform ordering for all ranks and accuracies. The improvement here is the combination of a sufficient residual bound and a nearly linear cost to obtain it with independence access.

The different dependence on accuracy also reflects how the prefix is accounted for. Ene–Nguyen choose $c = \Theta(1/\varepsilon)$ and preserve a compatible subset of the optimum with expected loss at most $\mathbb{E}f(S)/c$ (Ene and Nguyen, 2019, full version, Lemmas 12–13). We use the constant payment in (G.7): value gained by S pays for value lost by the comparison base C. Section H.6 shows why this is enough after adding S back to the fractional solution. The accuracy still controls the threshold grid and the later sampling; it does not require a leading $1/\varepsilon$ factor in our bound on ρ.

G.8 Known parameters and a bounded implementation

Set $L_* = \lceil \log_2(2 + \log(2n)) \rceil$ and choose the known output factor

$$\rho = \max\left\{1, \frac{C_1 \log^2(2n)\log(2r)}{1-\kappa}\right\}.$$

No optimization is required to compute this upper bound. Use a deterministic cap

$$Q_* = C_{\mathrm{cap}} n \kappa^{-2} \log^{A_{\mathrm{cap}}} \frac{2n}{\kappa\zeta}$$

on preprocessing queries and calls, where the absolute constants exceed the bound just proved, and set $\alpha = \zeta/(10Q_*)$ for every density call. The good-event bound depends on α only logarithmically. Increasing A_{cap} and then C_{cap} absorbs $\log Q_*$ and makes this a fixed, noncircular implementation choice.

Conditionally on the history before each of at most Q_* calls, Lemma G.5 fails with probability at most α. A union bound therefore gives total failure probability at most $\zeta/10$. On the complementary event the complexity bound prevents a cap abort. An exhausted cap or a failed search is observable and returns the current prefix with status abort. An unsound pilot is not observable; it does not create an extra branch in the algorithm.

For the comparison bound, condition on the current history and pilot, then average the independent search and basis-exchange coupling. Sound acceptances satisfy Lemma G.14; each unsound acceptance can instead lose at most OPT from the comparison base. The expected number of these acceptances is at most $Q_*\alpha$, using the sum of conditional error probabilities. Hence their total additive contribution to (G.7) is $O(\zeta\mathrm{OPT})$, independent of rank. An abort stops this coupling at its current feasible prefix. The final composition in Section H.6 uses only nonnegativity on bad executions and never assumes that the algorithm can detect them all.

G.9 Implementing the density test and proving its value payment

ProcessBases has so far used the accept-or-filter guarantee of Lemma G.5. We now implement that local decision; nothing in the threshold loop or recursion changes here. The input is a candidate set A independent in M/S, a fixed prefix S, and a threshold θ. We want either a uniformly chosen high element from at least h_0 choices, or exact low certificates leaving fewer than $2h_0$ high elements. A small pilot sample selects between fresh random search and a full scan. The count target and failure allowance are abstract inputs; preprocessing uses $h_0 = \lceil 4\Gamma/\theta \rceil$.

Algorithm G.4. DensityCheck$(A, S, \theta; h_0, \alpha)$

Output: $\mathrm{Accept}(e)$, $\mathrm{Filter}(D)$, or a detected abort.

$m \leftarrow |A|$.
if $m < h_0$ **then**
  return $\mathrm{Filter}(\{e \in A : f(e \mid S) < \theta\})$, found by a full scan.
end if
Draw $s = \lceil 48(m/h_0)\log(4/\alpha) \rceil$ uniform pilot samples from A, independently with replacement, and evaluate their marginals.
if the observed high fraction is below $3h_0/(2m)$ **then**
  return $\mathrm{Filter}(\{e \in A : f(e \mid S) < \theta\})$, found by a full scan.
else
  Generate a fresh independent random ordering of A.
  for the first $\min\{m, \lceil (m/h_0)\log(4/\alpha) \rceil\}$ positions e **do**
    if $f(e \mid S) \geq \theta$ **then**
      return $\mathrm{Accept}(e)$.
    end if
  end for
  return abort.
end if

The pilot distinguishes fewer than h_0 high elements from at least $2h_0$ high elements. Either action is useful in the gap between them. An independent random ordering makes the returned high element uniform; a full scan instead supplies exact low certificates. Certified-low elements remain low as S grows, by diminishing returns.

Proof of Lemma G.5. Keep S fixed throughout the call and let $H = \{e \in A : f(e \mid S) \geq \theta\}$. If $m < h_0$, the exact scan proves the statement. Otherwise let X count high pilot samples; X is binomial with mean $s|H|/m$. If $|H| < h_0$, declaring dense requires $X \geq 3sh_0/(2m)$. The upper-tail Chernoff bound, with stochastic domination by a binomial of mean sh_0/m, bounds this probability by $\exp(-sh_0/(12m)) \leq \alpha/4$. If $|H| \geq 2h_0$, declaring sparse requires a downward deviation of at least one quarter of the mean. The lower-tail Chernoff bound gives $\exp(-sh_0/(16m)) \leq \alpha/4$. These are standard bounded-sum inequalities (Boucheron et al., 2013, Chapter 2). If the dense cutoff exceeds one, dense declaration is impossible and $|H| < 2h_0$ is automatic.

On a sound dense declaration, $|H| \geq h_0$. For a fresh uniform ordering, any permutation of labels within H preserves the law. Its first high element is therefore uniform in H, even after conditioning on finding one before the cap, because that event also preserves the same symmetry. The chance that k inspected positions contain no high element is at most $(1 - |H|/m)^k \leq e^{-kh_0/m}$. Taking the specified cap makes this at most $\alpha/4$; inspecting all m positions succeeds whenever $H \neq \varnothing$. The pilot and search failure bounds sum to less than α. Finally, all returned low certificates and the accepted marginal are checked exactly. The stated query costs count the pilot, search, and possible full scan. □

Example G.13 (The two outcomes of a density check)**.** With $m = 100$ and $h_0 = 10$, 30 high elements allow a sound call to accept uniformly among those 30. Six high elements instead make a sound call scan and certify the other 94 as low. With 15 high elements, either outcome is useful: acceptance has at least ten choices, and filtering leaves fewer than twenty high elements.

G.9.1 Why uniform acceptance preserves value

Lemma G.14 (Uniform high-density extraction)**.** *Let S be independent, let A be independent in M/S, and let C be a base containing S. Fix any nonempty set $H \subseteq \{e \in A : f(e \mid S) \geq \theta\}$ of size h. If e is uniform in H, there is an analytical coupling to $c \in C \setminus S$ such that $C - c + e$ is a base containing $S + e$ and*

$$\mathbb{E}[f(C) - f(C - c + e)] \leq \frac{f(C) - f(S)}{h} \leq \frac{\mathrm{OPT}}{h}.$$

In particular, if $\theta h \geq (\mathrm{e} - 1)\mathrm{OPT}$, the accepted gain is at least $\mathrm{e} - 1$ times the expected comparison loss.

Proof. Extend A to a base of M/S. Between the resulting original base and C, use a basis-exchange bijection that fixes their common elements. This is the classical bijective basis-exchange fact (Oxley, 2011). Restrict it to H; its images are h distinct elements $D \subseteq C \setminus S$. For clarity, the one-sided version of the bijection follows from Hall's theorem. For a set Y of entrants outside the common part, let $N(Y)$ be all eligible deletions outside the common part. Every entrant lies in the closure of the common part together with $N(Y)$, by its fundamental circuit. Independence of the common part together with Y implies $|Y| \leq |N(Y)|$. Hall's condition gives a matching; fix the common elements to complete the bijection. Thus a uniform entrant has a uniform image in D, even if H was determined by earlier value queries.

For a matched pair, monotonicity gives $f(C) - f(C - c + e) \leq f(C) - f(C - c)$. Enumerate D in any order and delete its elements successively. Each successive deletion has marginal at least its deletion marginal from the full set C, by submodularity. Hence

$$\sum_{c \in D} (f(C) - f(C - c)) \leq f(C) - f(C \setminus D) \leq f(C) - f(S).$$

Average over D. Every accepted element gains at least θ, which proves the final assertion. Neither C nor the matching is computed. □

Example G.15 (Why uniformity is part of the algorithm)**.** Suppose a candidate independent set has $h = 20$ high elements. The total removal-marginal mass on their matched optimum positions is at most OPT. A uniform choice therefore loses at most $\mathrm{OPT}/20$ on average. Choosing a particular high element after inspecting objective values could concentrate on the one position responsible for almost all removal mass. The proof uses an average over distinct matched positions, so merely choosing an arbitrary high survivor would not justify it.

G.10 Supporting scale estimate

The complete fractional algorithm calls **EstimateScale** once before **Precondition**. It supplies Γ so that the threshold grid and the required number of high choices can be set without knowing OPT. It receives the original oracles and returns only a constant-factor upper bound; the greedy set used to obtain that number is not committed to S. We use the standard threshold-greedy construction below.

Lemma G.16 (EstimateScale). *For a normalized objective after removing loops,* **EstimateScale** *returns* $\mathrm{OPT} \le \Gamma \le 3\mathrm{OPT}$ *in* $O(n\log(2r))$ *value and independence queries (Buchbinder et al., 2017, Lemma 3.3). If the rank is zero, it returns zero.*

Proof. If all feasible singleton values are zero, zero is optimal by submodularity. Otherwise a standard decreasing threshold greedy procedure gives an independent set S_0 with $f(S_0) \ge \mathrm{OPT}/3$ using $O(n\log(2r))$ queries. Thus

$$\Gamma = 3f(S_0), \qquad \mathrm{OPT} \le \Gamma \le 3\mathrm{OPT} \tag{G.8}$$

is available. The following short proof records a concrete version of the standard constant-factor threshold argument; compare (Buchbinder et al., 2017, Lemma 3.3).

Start at the maximum feasible singleton θ_0, decrease thresholds by $7/8$, and stop at the first threshold at most $\theta_0/(8r)$. Scan every unselected feasible element at every threshold and accept it if its current marginal reaches the threshold. Pad the accepted sequence by dummies and match an optimum base to it using basis exchange, always removing an unmatched optimum position. A position matched at a later threshold was feasible and unselected in the previous pass, so its final marginal is at most the previous threshold. A dummy match is charged the final threshold. If W is the sum of thresholds of accepted elements, this gives

$$\mathrm{OPT} - f(S_0) \le \frac{8}{7}W + \frac{1}{8}\mathrm{OPT}, \qquad W \le f(S_0).$$

Consequently $f(S_0) \ge 49\mathrm{OPT}/120 > \mathrm{OPT}/3$. The exchange matching is described fully in the continuous-phase proof in Section H; the same deterministic argument applies here. □

The estimate does not commit the greedy set S_0 to the objective. **Precondition** starts its own prefix at $S = \varnothing$.

H Independence access: fast fractional solutions

H.1 Spend the range bound produced by preprocessing

The main new work in the fractional phase is behind us. The remaining step adapts decreasing-threshold continuous greedy (Badanidiyuru and Vondrák, 2014, Section 4) with singleton-proportional sampling allowances (Buchbinder et al., 2017, Section 3.1 and Corollary 3.2). We explain the connection to preprocessing first, then give the full implementation and proof.

The prefix S is now fixed. Preprocessing has changed the residual problem so that its marginal estimates are cheap; this phase uses those estimates to construct the fractional solution. Each output base D_s lies in M/S, and the full algorithm returns the bases $S \cup D_s$ to the rounder. Thus the prefix's value is retained in every possible rounded solution. The task left here is to find good residual directions and record them as a short base list.

Pay for a larger ρ with samples. A marginal sample for element e lies between zero and its residual singleton value $a_e = g(\{e\})$. We use the singleton-proportional allowances of (Buchbinder et al., 2017, Section 3.1), setting $b_e = a_e/\rho$. For every feasible direction or comparison set I,

$$\sum_{e\in I} \nu b_e = \frac{\nu}{\rho}\sum_{e\in I} a_e \le \nu\Lambda.$$

Thus a larger range factor need not cause a larger approximation loss: we divide the individual additive allowances by that factor. The cost is more samples. Lemma H.2 estimates a marginal μ_e to error at most $\nu\mu_e + \nu a_e/\rho$ using $O(\rho\nu^{-2}\log(2B/\xi))$ samples per test, where B bounds the number of tests and ξ is their total failure allowance. The mixed relative and additive error bound makes this dependence linear in ρ.

Consequently, Theorem H.6 uses $\widetilde{O}(N\rho\nu^{-4})$ value queries. The number of phases, $T = \lceil 1/\nu \rceil$, and the bound on independence queries have no multiplicative ρ factor. The polylogarithmic factor obtained in Section G.7 is therefore sufficient: with $\nu = \Theta(\varepsilon)$, the continuous phase remains nearly linear in N. The approximation loss stays $O(\nu\Lambda)$, rather than growing to $O(\nu\rho\Lambda)$.

The input is a residual matroid M_0 on N elements, a normalized monotone submodular objective g, and numbers Λ, ρ satisfying

$$\mathrm{OPT}_g \le \Lambda, \qquad a_e = g(\{e\}), \qquad \max_{I \in \mathcal{I}(M_0)} \sum_{e \in I} a_e \le \rho\Lambda, \qquad \rho \ge 1. \tag{H.1}$$

Write G for the multilinear extension of g.

Distinguish the original and residual optimum scales. The usual range-sensitive guarantee is stated with a bound of the form $\max_I \sum_{e \in I} a_e \le c\mathrm{OPT}_g$ (Buchbinder et al., 2017, Corollary 3.2). Our condition (H.1) allows Λ to exceed OPT_g: in the application $\Lambda = \Gamma$ is comparable to the original optimum, whereas much of that value may already be in S. For example, in a free matroid on two elements with modular values 99 and 1, a prefix containing the first leaves $\mathrm{OPT}_g = 1$, although $\Lambda = 100$ is a valid original optimum scale. For a fixed coefficient, a bound against the larger scale is weaker. We therefore state the effect of this normalization explicitly.

Theorem H.6 guarantees

$$G(x) \ge (1 - 1/e - 4\nu)\mathrm{OPT}_g - 3\nu\Lambda$$

with the stated high probability. The additive term may be large relative to a tiny residual optimum, but it is only $O(\nu\mathrm{OPT})$ for the full problem, since $\Lambda = \Gamma \le 3\mathrm{OPT}$. The final solution retains $f(S)$, and the prefix payment from preprocessing compensates for the comparison value lost by fixing S; Section H.6 proves the resulting $1 - 1/e - \varepsilon$ guarantee. We do not need to assume that the residual optimum remains a constant fraction of the original one. The continuous-greedy mechanism is the established one; the explicit allowances and scale term make it apply to the bound our preprocessing actually supplies.

The output is a short list of residual bases. In each phase, decreasing thresholds select an independent direction from estimated marginals. Every acceptance immediately increases that coordinate of the fractional point. Later estimates use the new point. After all phases we extend the saved directions to bases and return their average.

Example H.1 (How a support-scaled allowance is spent)**.** In a rank-two uniform matroid, let the three elements cover the nested feature sets $\{1, 2, 3\}$, $\{1, 2\}$, and $\{1\}$, with unit feature weights. Then $a = (3, 2, 1)$, $\mathrm{OPT}_g = \Lambda = 3$, and one may take $\rho = 5/3$. Thus $b = (1.8, 1.2, 0.6)$, and the sum of the two largest allowances is exactly Λ. The individual estimates have different additive tolerances νb_e, but the proof can spend at most $\nu\Lambda$ on any direction or comparison base. For a rank-r family of duplicate elements covering one feature, the unconditioned ratio can instead be $\rho = r$; accepting one such element in preprocessing leaves zero residual singleton ranges. This explains the purpose of the prefix.

Algorithm H.1. SupportedFractional$(M_0, g, \Lambda, \rho; \nu, \xi)$

Input: (H.1), $0 < \nu \le 1/8$, $0 < \xi < 1$.

Delete loops by singleton independence tests; compute r_0 by greedy extension; evaluate feasible $a_e = g(\{e\})$.
if the rank $r_0 = 0$ or all feasible singleton values vanish **then**
return one arbitrary base of M_0.
end if
Keep positive-singleton elements active. Set $a_{\max} = \max a_e$.
$T \leftarrow \lceil 1/\nu \rceil$; $\delta \leftarrow 1/T$; $x \leftarrow 0$.
Set $\tau_j = (1-\nu)^j a_{\max}$, and let J be the first index with $\tau_J \le \nu a_{\max}/r_0$. Set $B \leftarrow NT(J+1)$.
for $s = 1, \dots, T$ **do**
$I_s \leftarrow \varnothing$; $y \leftarrow x$.
for $j = 0, \dots, J$ **do**
for each active $e \notin I_s$, in a fixed order **do**
if $I_s + e \in \mathcal{I}(M_0)$, tested by the independence oracle **then**
$\widehat{\mu} \leftarrow$ **EstimateMarginal**$(g, e, y, a_e; \rho, \nu, \xi/B)$.
if $\widehat{\mu} \ge \tau_j$ **then** $I_s \leftarrow I_s + e$ and immediately $y \leftarrow y + \delta \mathbf{1}_e$.
end if
end for
end for
$x \leftarrow y$; save I_s.
end for
for $s = 1, \dots, T$ **do**
Greedily extend I_s to a base D_s of M_0, restoring zero-singleton elements if needed.
end for
return $D_1, \dots, D_T$.

Inside a phase, the independence oracle answers whether the next element fits in the current direction. Only then does **EstimateMarginal** answer the value question: is its derivative large enough for this threshold? The estimate is a number used by the caller, not a new fractional solution. An acceptance immediately changes the fractional point, so the next value test samples at that updated point. Saving the phase's accepted set records the direction we have just added; extending it to a base at the end gives the representation needed by rounding.

The proof follows the same sequence. First it bounds the estimator's error using the range allowance. Then it relates each accepted coordinate update to its exact finite gain. Finally it matches optimum elements to past rejections to show that these gains suffice for continuous greedy. The explicit base-list representation follows the established framework (Kobayashi and Terao, 2024, Section 3); no new decomposition algorithm is needed.

H.2 EstimateMarginal: answer one threshold test

Put $b_e = a_e/\rho$. Remove loops and elements with $a_e = 0$ from the active optimization; their derivatives vanish everywhere by monotonicity and DR. Zero-singleton elements can be restored when extending independent sets to bases; loops are never used. Removing loops before taking $a_{\max}$ also ensures $a_{\max} \le \mathrm{OPT}_g$. At a point $y \in [0,1]^N$, a sample for $\partial_e G(y)$ is

$$X = g(R \cup \{e\}) - g(R), \qquad R \subseteq E \setminus \{e\}, \quad \mathbb{P}[d \in R] = y_d \text{ independently.}$$

Then $0 \le X \le a_e$ and $\mathbb{E}X = \mu = \partial_e G(y)$.

Lemma H.2 (**EstimateMarginal** and simultaneous accuracy)**.** *For $0 < \nu \le 1/8$, B adaptively chosen tests, and failure probability ξ, using*

$$m = \left\lceil C \rho \nu^{-2} \log \frac{2B}{\xi} \right\rceil$$

fresh samples per test, for a sufficiently large absolute C, and returning their mean $\widehat{\mu}_e(y)$ guarantees simultaneously

$$|\widehat{\mu}_e(y) - \partial_e G(y)| \le \nu \partial_e G(y) + \nu b_e \tag{H.2}$$

with probability at least $1-\xi$. One call with failure allowance ω takes $O(\rho\nu^{-2}\log(2/\omega))$ ordinary value queries and returns the estimate in this bound.

Proof. Normalize $U = X/a_e \in [0,1]$ and set $u = \mathbb{E}U$. Its variance is at most u. The bounded-variable Bernstein inequality (Boucheron et al., 2013, Chapter 2) gives

$$\mathbb{P}[|\overline{U} - u| > t] \le 2\exp\left(-\frac{mt^2}{2u + 2t/3}\right).$$

Take $t = \nu(u + 1/\rho)$. Since $\nu \le 1$, $2u + 2t/3 \le (8/3)(u + 1/\rho)$, whereas $t^2 = \nu^2(u + 1/\rho)^2$. The exponent is at least $3m\nu^2/(8\rho)$. Multiplication by a_e gives the desired error. For an adaptive test, condition on its entire past. The point y, the coordinate, and the range are fixed, and the new samples are independent of that past. The same failure bound therefore holds conditionally. Summing these conditional bounds over at most B tests proves the claim. This use of a standard bounded-variable inequality does not require that future adaptive queries be fixed in advance. □

H.3 Feasibility, the test budget, and exact accepted gain

Lemma H.3 (Finite implementation)**.** *Every run of Algorithm H.1 stays in the unit cube and returns bases representing a feasible fractional point. It executes at most $B = NT(J+1)$ marginal tests and the same number of direction-feasibility tests, plus $O(NT)$ queries for base extensions.*

Proof. At the start of phase s, $x = \delta \sum_{i<s} \mathbf{1}_{I_i}$. Each I_i is independent because every addition was tested. During phase s, $y = x + \delta \mathbf{1}_{I_s}$ with I_s independent. Before an element is accepted, it is not in I_s, so its coordinate is at most $(s-1)\delta \le 1 - \delta$. Its update is inside the cube. Moreover y is a convex combination of at most s independent-set indicators and zero, since $s\delta \le 1$; thus $y \in P(M_0)$ as well.

There are T phases, $J+1$ thresholds, and at most N candidate checks per threshold. An infeasible candidate causes no value sampling. This proves the deterministic budget, regardless of sampling outcomes. Greedy extension of I_s takes at most N additional independence queries and yields a base $D_s \supseteq I_s$ by augmentation. Therefore the returned average is in the base polytope and dominates the pre-extension average coordinatewise. If all feasible singleton values are zero, DR implies $g(I) \le \sum_{e \in I} g(\{e\}) = 0$ on every independent set, so the early return has the same guarantee. □

H.4 Matching optimum elements to past rejections

Fix one phase and denote its beginning and end by x and y^+. For each accepted element let $\tau(e)$ be its threshold and $y^{(e)}$ the point immediately before its update. Define

$$W = \sum_{e \in I} \tau(e), \qquad D = \sum_{e \in I} \partial_e G(y^{(e)}).$$

On the event (H.2), acceptance implies $\tau(e) \le (1+\nu)\partial_e G(y^{(e)}) + \nu b_e$. Since I is independent, (H.1) gives

$$W \le (1+\nu)D + \nu\Lambda. \tag{H.3}$$

Also, by one-coordinate affinity,

$$G(y^+) - G(x) = \delta D. \tag{H.4}$$

Example H.4 (A finite update must use the current point)**.** For two elements covering the same unit-weight feature, $G(x) = 1 - (1-x_1)(1-x_2)$. At $x = (0.2, 0.3)$ the value is 0.44. With $\delta = 0.1$, increasing coordinate one gains $0.1(0.7) = 0.07$. The new point is $(0.3, 0.3)$, so the derivative for coordinate two is now 0.7, and that update gains another 0.07. The final value at $(0.3, 0.4)$ is 0.58, exactly as (H.4) predicts. Using both initial derivatives would instead predict gain $0.1(0.7 + 0.8) = 0.15$. Immediate updates avoid having to pay for that accumulated discrepancy in a rank-dependent discretization estimate.

Lemma H.5 (Past rejections control a comparison optimum)**.** *On the simultaneous accuracy event, for an optimum independent set O,*

$$\mathrm{OPT}_g - G(y^+) \le \sum_{o\in O} \partial_o G(y^+) \le \frac{W}{(1-\nu)^2} + \frac{2\nu\Lambda}{1-\nu}. \tag{H.5}$$

Proof. **Step 1: turn the optimum gap into marginal values.** Increasing the coordinates of O to one gives $G(y^+ \vee \mathbf{1}_O) \ge g(O) = \mathrm{OPT}_g$. Add these coordinate increments one at a time. Each has length at most one and, by DR, derivative no larger than its derivative at y^+. Consequently $\mathrm{OPT}_g - G(y^+) \le \sum_{o\in O} \partial_o G(y^+)$. It remains to identify a useful earlier test for each such derivative.

Step 2: construct a matching with the needed feasibility property. Let r_0 be the residual rank. Use the rank-r_0 dummy extension of Section A, giving dummy elements zero objective value. Fix an optimum base O in that extension. Order the real elements accepted in this phase by their acceptance times and append enough distinct dummies to make a base. We maintain an analytical base C, initially O, that contains the selected prefix A and whose remaining elements are exactly the unmatched elements of the original O.

For the next selected element v, if $v \in C$, it is outside A and is an unmatched optimum element: match it to itself and regard it as part of the prefix. If $v \notin C$, the fundamental circuit in $C + v$ contains an element $o \in C \setminus A$. Otherwise it would be a circuit contained in $A + v$, contradicting feasibility of that selected prefix. Delete such an o, insert v, and match v to o. The new set is a base, the prefix grows by one, and exactly one unmatched optimum element is removed. Induction gives a bijection between all selected positions and O.

Immediately before o's partner is selected, $A + o$ is independent because it is a subset of the comparison base. Every earlier selected prefix is a subset of A, so o was feasible at every earlier test time. It was also unselected: an optimum element selected earlier would already have been matched and could not remain among the unmatched elements. These facts hold even when O and the algorithm's direction overlap.

Step 3: use the preceding complete threshold pass. Suppose o is matched to a real element accepted at τ_j with $j \ge 1$. If $a_o = 0$, its derivative is zero and no bound is needed. Otherwise o was an active, unselected, feasible candidate in pass $j-1$. The algorithm therefore queried its marginal at some point y' and rejected it by value: $\widehat{\mu}_o(y') < \tau_{j-1}$. All subsequent updates are nonnegative, so $y' \le y^+$ and

$$(1-\nu)\partial_o G(y^+) \le (1-\nu)\partial_o G(y') \le \widehat{\mu}_o(y') + \nu b_o < \tau_{j-1} + \nu b_o.$$

Since $\tau_{j-1} = \tau_j/(1-\nu)$, this gives

$$\partial_o G(y^+) \le \frac{\tau_j}{(1-\nu)^2} + \frac{\nu b_o}{1-\nu}.$$

If the partner was selected in the first pass, there is no preceding pass. Use $\partial_o G(y^+) \le a_o \le a_{\max} = \tau_0$, which is no larger than its allotted threshold term. Dummy optimum elements have zero derivative.

Step 4: handle positions left unfilled by the real direction. If o is matched to a final dummy position, it remains feasible and unselected throughout the last pass. A positive-singleton o therefore has a last failed value test, giving $\partial_o G(y^+) \le (\tau_J + \nu b_o)/(1-\nu)$ by the same argument. Zero-singleton elements again contribute zero. There are at most r_0 such positions, and $r_0\tau_J \le \nu a_{\max} \le \nu\Lambda$.

Step 5: add the bounds along the matching. The real partners contribute $W/(1-\nu)^2$ in threshold terms. Final positions contribute at most $\nu\Lambda/(1-\nu)$. Since the matching uses each optimum element once and O is independent, $\sum_{o\in O} b_o \le \Lambda$, so all range allowances contribute at most another $\nu\Lambda/(1-\nu)$. This proves (H.5). The matching is analytical; the algorithm never needs O or C. □

H.5 Solving the recurrence and returning explicit bases

Theorem H.6 (Supported continuous phase)**.** *Under* (H.1)*, Algorithm H.1 returns an explicit average of $T = \lceil 1/\nu \rceil$ independent sets whose extension to bases has multilinear value at least*

$$(1 - 1/e - 4\nu)\mathrm{OPT}_g - 3\nu\Lambda$$

with probability at least $1 - \xi$. It uses

$$O\big(N\rho\nu^{-4}\,\mathrm{polylog}(Nr/(\nu\xi))\big) \text{ value queries}, \qquad O\big(N\nu^{-2}\log(2r/\nu)\big) \text{ independence queries}.$$

Proof. Combining (H.3) and (H.5) and solving for D yields, with $a = (1-\nu)^2/(1+\nu)$,

$$D \geq a(\mathrm{OPT}_g - G(y^+)) - \left(\frac{\nu}{1+\nu} + \frac{2\nu(1-\nu)}{1+\nu}\right)\Lambda.$$

Here $1 - 3\nu \leq a \leq 1$, and $G(y^+) \geq 0$. Thus

$$D \geq (1 - 3\nu)\mathrm{OPT}_g - G(y^+) - 3\nu\Lambda.$$

Using (H.4) and writing G_s for the value after phase s,

$$(1+\delta)G_s \geq G_{s-1} + \delta\big((1-3\nu)\mathrm{OPT}_g - 3\nu\Lambda\big).$$

Since $G_0 = 0$, induction gives

$$G_T \geq c_T\big((1-3\nu)\mathrm{OPT}_g - 3\nu\Lambda\big), \qquad c_T = 1 - (1+\delta)^{-T}.$$

For $0 < \delta \leq \nu$, $T\log(1+\delta) \geq 1 - \delta/2$, so $c_T \geq 1 - e^{-1+\delta/2} \geq 1 - 1/e - \nu$. The last estimate follows, for example, by the mean-value theorem on $[-1, -1+\nu/2]$. As $0 \leq c_T \leq 1$, the preceding display is at least $(1 - 1/e - 4\nu)\mathrm{OPT}_g - 3\nu\Lambda$. This bound remains valid if the expression in parentheses is negative: its negative Λ term is bounded using $c_T \leq 1$ separately.

Extend each I_s greedily to a base of the original residual matroid, restoring deleted zero-marginal elements as needed. At most NT additional independence queries suffice. The averaged point increases coordinatewise, so its multilinear value cannot decrease. For the query count, $T = O(\nu^{-1})$ and $J + 1 = O(\nu^{-1}\log(2r_0/\nu))$. Thus $B = O(N\nu^{-2}\log(2r_0/\nu))$. Multiplying by $O(\rho\nu^{-2}\log(2B/\xi))$ value queries per estimate proves the stated value bound; direction checks and extensions give the independence bound. Here r_0 is the residual rank (denoted r in the displayed bounds). All reasoning on the simultaneous event is pathwise; we do not condition a fresh estimator on a future success event. □

H.6 Combining preprocessing and fractional optimization

Define **NearlyLinearFractional** as follows. Remove loops, compute the rank by a greedy scan, obtain Γ from Lemma G.16, and return an arbitrary base if $\Gamma = 0$. Otherwise run **Precondition**. On a detected abort, return one greedy base extending its prefix S. On completion, run **SupportedFractional** on M/S and $g(I) = f(S \cup I) - f(S)$, with $\Lambda = \Gamma$ and the returned ρ. Add S to every returned residual base. The resulting list represents the full fractional point; the proof below specifies the accuracy parameters and accounts for all failure events.

Theorem H.7 (Nearly linear fractional optimization with independence access)**.** *For every $0 < \varepsilon < 1/2$, a normalized monotone submodular objective over a rank-r matroid admits an explicit convex combination of $O(1/\varepsilon)$ bases with expected multilinear value at least $(1 - 1/e - \varepsilon)\mathrm{OPT}$, using $n\,\mathrm{poly}(1/\varepsilon, \log(2n))$ value and independence queries.*

Why the prefix payment is exactly the right one. Let $c = 1 - 1/e$ and temporarily ignore small error terms. The comparison base $C \supseteq S$ provides a residual feasible set of value $f(C) - f(S)$. Adding the prefix to a c-approximate residual solution gives $cf(C) + (1-c)f(S)$. A comparison loss $L = \mathrm{OPT} - f(C)$ costs cL in this expression. The prefix supplies $(\mathrm{e} - 1)L$ in expectation, and $(1-c)(\mathrm{e}-1) = c$. Thus its gained value pays exactly for that loss. The following proof includes finite accuracy and all failure events.

Proof. **Step 1: fix implementable parameters.** In **NearlyLinearFractional**, set

$$\kappa = \nu = \xi = \varepsilon/100, \qquad \zeta = \frac{\varepsilon}{100(K_{\mathrm{cmp}} + 2)}.$$

Use the depth, density failure allocation, and known ρ of Section G.8. The cap is part of the preprocessing algorithm. An observable abort returns a greedy base extending S; an undetected bad pilot does not cause a hypothetical extra branch. All executions return a feasible base list with nonnegative value.

Step 2: condition only on the completed preprocessing history. Let $\mathcal{A}$ be the event that preprocessing completes with sound certificates. Its probability is at least $1 - \zeta$. Include the analytical comparison coupling in the history, so S, C, Γ are fixed when conditioning. On $\mathcal{A}$, the residual objective obeys (H.1) with $\Lambda = \Gamma$, and $\mathrm{OPT}_g \geq f(C) - f(S)$. Put $a = 1 - 1/e - 4\nu$ and

$$H = af(C) + (1 - a)f(S) - 3\nu\Gamma.$$

Theorem H.6, applied with fresh future sampling, gives final full value at least H except with conditional probability at most ξ. Also $H \leq \mathrm{OPT}$, because $0 \leq a \leq 1$ and both S, C are feasible. If $H \geq 0$, nonnegativity on sampling failure gives conditional expected value at least $(1 - \xi)H \geq H - \xi\mathrm{OPT}$. If $H < 0$, the same lower bound follows directly from nonnegativity. This treats negative lower bounds without multiplying an inequality by an inappropriate success probability.

Step 3: remove the preprocessing-event restriction. The same expression H is defined on every preprocessing history through its stopped comparison base, including aborted histories. Although the residual budget need not hold off $\mathcal{A}$, $H \leq \mathrm{OPT}$ still holds. Writing X for the returned fractional point and using nonnegativity off $\mathcal{A}$ gives

$$\mathbb{E}F(X) \geq \mathbb{E}[\mathbf{1}_{\mathcal{A}} H] - \xi\mathrm{OPT} \geq \mathbb{E}H - (\xi + \zeta)\mathrm{OPT}.$$

This is an analytical restriction of the expectation, not an algorithmic attempt to recognize every sampling error.

Step 4: use the prefix payment and collect the losses. Let $L = \mathrm{OPT} - f(C) \geq 0$. Theorem G.12 gives $\mathbb{E}f(S) \geq (\mathrm{e} - 1)\mathbb{E}L - K_{\mathrm{cmp}}\zeta\mathrm{OPT}$. Since $\Gamma \leq 3\mathrm{OPT}$, substitution yields

$$\begin{aligned}\mathbb{E}F(X) &\geq a\mathrm{OPT} + \big((1 - a)(\mathrm{e} - 1) - a\big)\mathbb{E}L - 9\nu\mathrm{OPT} - \big(\xi + \zeta + (1 - a)K_{\mathrm{cmp}}\zeta\big)\mathrm{OPT} \\ &\geq \big(1 - 1/e - 13\nu - \xi - (K_{\mathrm{cmp}} + 1)\zeta\big)\mathrm{OPT}.\end{aligned}$$

The discarded coefficient is $(1 - a)(\mathrm{e} - 1) - a = 4\mathrm{e}\nu \geq 0$. The chosen parameters make the total loss less than $\varepsilon\mathrm{OPT}$.

Step 5: verify the returned representation and cost. Every D_s is a base of M/S, so $S \cup D_s$ is an original base. Furthermore the full multilinear value equals the residual multilinear value plus $f(S)$, since every prefix coordinate is one. There are $O(1/\nu) = O(1/\varepsilon)$ bases; early termination returns one base. Scale estimation, capped preprocessing, and the residual phase all cost $n\,\mathrm{poly}(1/\varepsilon, \log(2n))$ queries because $r \leq n$ and $\rho = \mathrm{polylog}(n)$. This proves the theorem. □

H.7 Return an integral solution

Run **NearlyLinearFractional** with accuracy $\varepsilon/2$ and pass its uniformly weighted base list to **IndependenceRound** with allowed loss $\varepsilon/2$. Conditionally on that list, rounding preserves a $1 - \varepsilon/2$ fraction of its value. The tower property and Theorem H.7 give

$$\mathbb{E}f(R) \geq (1 - \varepsilon/2)(1 - 1/e - \varepsilon/2)\mathrm{OPT} \geq (1 - 1/e - \varepsilon)\mathrm{OPT}.$$

The rounder adds $O(r\varepsilon^{-3}\log(2r))$ expected independence queries and no value queries. Since $r \leq n$, this proves the independence-oracle part of Theorem C.1.

I POLYMATROID OPTIMIZATION WITH DOUBLY LOGARITHMIC RANK DEPENDENCE

I.1 THE RESULT AND THE BARRIERS TO NEARLY LINEAR COMPLEXITY

Let $E = [n]$ and let $\rho : 2^E \to \mathbb{Z}_+$ be normalized, monotone, and submodular. Its integral polymatroid is

$$P = P(\rho) = \{x \in \mathbb{R}_+^E : x(S) \le \rho(S) \quad (S \subseteq E)\}.$$

Write $r = \rho(E)$ and $u_e = \rho(\{e\})$. The objective $f : \mathbb{Z}_+^E \to \mathbb{R}_+$ is normalized, monotone, and satisfies

$$\Delta_e f(x) := f(x + \mathbf{1}_e) - f(x) \ge \Delta_e f(y) \quad \text{whenever } x \le y.$$

We have an exact ordinary value oracle at every nonnegative integer argument, including infeasible ones, and a one-bit integer-feasibility oracle for P. No rank, separation, or real-point membership oracle is assumed. The rank and singleton capacities need not be supplied.

Theorem I.1 (Polymatroid guarantee)**.** *For $0 < \varepsilon \le 1/2$, there is an algorithm returning $X \in P \cap \mathbb{Z}_+^E$ on every run such that*

$$\mathbb{E}f(X) \ge (1 - 1/e - \varepsilon)\mathrm{OPT}, \qquad \mathrm{OPT} = \max_{x \in P \cap \mathbb{Z}_+^E} f(x).$$

With $\Xi_\varepsilon = \log(2n/\varepsilon) + \log\log(r+3)$, it uses

$$Q_f = O(n\varepsilon^{-9}\Xi_\varepsilon^3), \qquad \mathbb{E}Q_P = O(n\varepsilon^{-11}\Xi_\varepsilon^3)$$

original oracle calls. In particular, when $r = 2^{\Theta(n)}$, both costs are $n\,\mathrm{poly}(1/\varepsilon, \log n)$.

For fixed accuracy and $r = 2^{\Theta(n)}$, this gives nearly linear oracle complexity in the number of coordinate types. Proposition I.26 proves an $\Omega(n)$ feasibility-query lower bound even at rank exactly 2^n. Thus the dimension dependence is tight up to polylogarithmic factors in that regime. Numerical bit complexity and the optimal accuracy dependence remain separate questions.

Three costs survive ordinary count batching. Counts avoid expanding one coordinate into all its units, but three different uses of numerical rank can still be expensive. A global threshold scan traverses a range of size polynomial in r; an exact count search takes $\Theta(\log r)$ endpoint tests; and concentrating every optimum unit separately introduces $\log r$ into sample size. The construction below addresses all three within one algorithm. Random separation of capacity scales and a small integer baseline shorten the threshold range. Geometric count searches use only $O(\log\log r)$ tests, with shadow counts repairing their comparison certificates. Concentration of a weighted aggregate of latent errors removes the union bound over optimum units. Poisson smoothing and colors then permit large fractional steps, and a small denominator keeps final rounding inexpensive.

I.2 THE WHOLE CONSTRUCTION BEFORE ITS SUPPORTING ARGUMENTS

Fix $h = \lceil 16/\varepsilon \rceil$. A random shift retains coordinate groups separated by gaps in their capacity exponents. Their direct-sum polymatroid Q satisfies $(1 - 1/h)Q \subseteq P$. An integer baseline a uses at most a $1/h$ fraction of every group's capacity. We optimize the normalized residual objective $g(x) = f(a + x) - f(a)$ over $Q_a = \{x \ge 0 : a + x \in Q\}$, preserving interactions between groups in every value query.

Inside each fractional step, h color vectors share one feasible direction $T \in Q_a$. Different colors use different value contexts; they have no separate capacity quotas. The direction is found by group thresholds and geometric count searches. After h^2 steps, one shrink places the fractional point in the original P, and the local rounder returns the final integer vector. The following algorithm specifies this flow; the remaining subsections define each quantity and prove its guarantee.

Algorithm I.1. ScaleSeparatedPolymatroid(f, P, ε)

Set $h = \lceil 16/\varepsilon \rceil$, $\eta = \kappa = 1/h$, and $L = h^2$.
Find capacity exponents by Lemma I.2; sample the shift and form retained groups.
If no types remain, return zero.
Construct a, $g(x) = f(a + x) - f(a)$, V_0, and each group's thresholds.
If $V_0 = 0$, return zero. Skip groups with no feasible residual singleton.
Bound all possible sample batches by B using the known finite grids.
Set $M = \lceil h^4 \log(4hB)/2 \rceil$ as in Lemma I.17, and $z = 0$.
for $i = 1, \ldots, L$ **do**
  Call $(T, q_1, \ldots, q_h) =$ **GroupedSampledSplit**(z, M) with fresh samples.
  Update $z \leftarrow z + T/L$.
end for
Set $y = (1 - 1/h)(a + z)$.
return LocalRound$(P, y, D = h^3, \xi = 1/h)$, using Algorithm I.4.

No optimum, rank value, shadow count, or real-point membership query is required by these operations.

I.3 Separate capacity scales, then bound residual marginals

I.3.1 A small random gap makes groups nearly independent

The purpose of the gaps is to control cross-group constraints without examining them individually. For any set of types, its highest retained scale pays for its own contribution; the total contribution from all lower groups is only an $O(1/h)$ fraction of a single highest-group capacity. Separation alone does not control marginal ranges. The integer baseline supplies that second property by diminishing returns.

Fix throughout

$$h = \lceil 16/\varepsilon \rceil, \qquad \gamma = 1/h.$$

In particular $h \geq 32$.

Lemma I.2 (Capacity exponents)**.** *With $O(n(1 + \log\log(r + 2)))$ original feasibility calls, one can identify zero-capacity types and find integers $k_e \geq 0$ for the others such that*

$$\ell_e := 2^{k_e} \leq u_e < 2^{k_e+1}, \qquad \bar{u}_e := 2^{k_e+1} - 1 \geq u_e.$$

Proof. Test $\mathbf{1}_e$ first. If feasible, find the least integer $a \geq 1$ such that $2^a \mathbf{1}_e$ is infeasible. Exponential search on the exponent $a = 1, 2, 4, \ldots$, followed by binary search on the bracketed exponent interval, costs $O(1 + \log\log(u_e + 2))$ calls. Set $k_e = a - 1$. No exact capacity is recovered. □

Remove zero-capacity types. If none remain, return zero. Define integer lengths

$$d = \lceil \log_2(nh) \rceil, \qquad p = hd, \qquad w = (h - 1)d.$$

Choose a uniform shift $s \in \{0, \ldots, p - 1\}$. Retain e precisely when $(k_e + s) \bmod p < w$, and group retained types by $\lfloor (k_e + s)/p \rfloor$. Write H for the retained types and $B_1, \ldots, B_m$ for the nonempty groups in increasing scale order. Only these at most n nonempty groups are stored; intervals with no coordinate are never enumerated.

Let P_b be the restriction of P to coordinates B_b, and let

$$Q = \bigoplus_{b=1}^{m} P_b$$

on H, extended by zero on $E \setminus H$. This is an integral polymatroid. It relaxes the cross-group constraints of P.

Lemma I.3 (Containment and retention)**.** *For every shift,*

$$P|_H \subseteq Q \subseteq (1+\gamma)P, \qquad (1-\gamma)Q \subseteq P.$$

For any fixed integer $O \geq 0$,

$$\mathbb{E}_s f(O|_H) \geq (1-\gamma) f(O).$$

Proof. The first inclusion follows by restriction. Take $x \in Q$ and any $S \subseteq E$ meeting H. Let B_b be the highest-scale group meeting $S \cap H$, and choose $e \in S \cap B_b$. All lower-group types a have $k_a \leq k_e - d - 1$, by the discarded gap between active intervals. Consequently

$$u_a < 2^{k_a+1} \leq 2^{k_e-d} \leq 2^{-d} u_e.$$

The contribution of the highest group is at most $\rho(S \cap B_b) \leq \rho(S)$. Each lower-group coordinate is at most its singleton capacity, so their combined contribution is at most $n2^{-d}u_e \leq \gamma\rho(S)$. Thus $x(S) \leq (1+\gamma)\rho(S)$, proving the second inclusion. The shrink assertion follows from $(1-\gamma)(1+\gamma) \leq 1$.

Every type is retained with probability $w/p = 1-\gamma$. Independence of retention events is not needed. Fix an order of the coordinate bundles of O, and let v_e be their successive marginal values, so $\sum_e v_e = f(O)$. By DR, for every retained set H, $f(O|_H) \geq \sum_{e \in H} v_e$: the earlier retained context is smaller than the full earlier context defining v_e. Take expectations. □

I.3.2 The baseline cancels the absolute capacity scale

For each nonempty group let

$$\ell_b = \min_{e \in B_b} \ell_e, \qquad t_b = \left\lfloor \frac{\ell_b}{hn} \right\rfloor, \qquad a_e = t_b \ (e \in B_b).$$

Set $a_e = 0$ on discarded types. Define

$$Q_a = \{x \geq 0 : a + x \in Q\}, \qquad V = f(a), \qquad g(x) = f(a+x) - V.$$

The residual polytope Q_a is an integral polymatroid and is the direct sum of $(P_b)_{a_b}$. This is the integral contraction operation. For a polymatroid with integral rank function σ, an integral feasible translation a changes rank to $\sigma_a(S) = \min_{U \supseteq S}(\sigma(U) - a(U))$; submodularity follows by taking the union and intersection of minimizing supersets. The function g is normalized monotone DR.

Lemma I.4 (Baseline feasibility and preservation)**.** *We have $a \in \gamma Q$ and $a \in P$. Set*

$$G_* = \max_{x \in Q_a \cap \mathbb{Z}^H} g(x), \qquad A = V + G_*.$$

For every original optimum O and every fixed shift,

$$A \geq (1-\gamma) f(O|_H).$$

Proof. For every nonempty $S \subseteq B_b$, $a(S) = |S| t_b \leq \ell_b/h \leq \gamma\rho(S)$. This proves $a \in \gamma Q$. By Lemma I.3, $a \in \gamma(1+\gamma)P \subseteq P$. Since $O|_H \in Q$ and $a/\gamma \in Q$, convexity gives $a + (1-\gamma)O|_H \in Q$. Thus $(1-\gamma)O|_H \in Q_a$. Its denominator divides h. By Lemma I.9, applied to g on Q_a, and the integer translation identity for local interpolation,

$$\begin{aligned} A &\geq V + F_g^{\mathrm{loc}}((1-\gamma)O|_H) \\ &= F_f^{\mathrm{loc}}(a + (1-\gamma)O|_H) \\ &\geq F_f^{\mathrm{loc}}((1-\gamma)O|_H) \geq (1-\gamma) f(O|_H). \end{aligned}$$

□

For every retained type test whether $\mathbf{1}_e \in Q_a$ using one original query at $a_b + \mathbf{1}_e$, zero outside its group. Evaluate $g(\mathbf{1}_e)$ for feasible residual singletons and set

$$V_0 = \max\big(\{V\} \cup \{g(\mathbf{1}_e) : \mathbf{1}_e \in Q_a\}\big).$$

This requires $O(n)$ original queries, including the evaluation of V. If $V_0 = 0$, DR implies $A = 0$, and zero may be returned for this shift. Otherwise $0 < V_0 \le A$. In particular V_0 is a known value scale; neither A nor G_* is evaluated.

Define the known residual-capacity upper bounds

$$U_e = \bar{u}_e - a_e \ge 0, \qquad R_b = \sum_{e \in B_b} U_e, \qquad \theta_b = \frac{V_0}{\max\{1, t_b\}}.$$

Empty residual groups can be skipped. Let r_b denote the unknown rank of $(P_b)_{a_b}$, so $r_b \le R_b$.

Lemma I.5 (Short local threshold ranges)**.** *Every feasible residual singleton in B_b has value at most θ_b. For $\kappa = 1/h$, use thresholds*

$$\tau_{b,j} = (1-\kappa)^j \theta_b$$

through the first K_b with $\tau_{b,K_b} \le \kappa V_0/(nR_b)$. Then

$$K_b + 1 = O(h^2 \log(2nh)), \qquad \sum_b r_b \tau_{b,K_b} \le \kappa V_0.$$

Proof. When $t_b = 0$ the singleton assertion follows from the definition of V_0. When $t_b > 0$, telescoping the t_b units of coordinate e in the baseline gives $f(a) \ge t_b \Delta_e f(a)$ by monotonicity and DR. Thus $g(\mathbf{1}_e) \le V/t_b \le \theta_b$.

All exponents in one retained group lie in an interval of w integer levels. Therefore $\bar{u}_e < 2^w \ell_b$ and $R_b < n 2^w \ell_b$. If $t_b \ge 1$, then $t_b \ge \ell_b/(2hn)$; if $t_b = 0$, then $\ell_b < hn$. In both cases,

$$\frac{R_b}{\max\{1, t_b\}} \le 2hn^2 2^w.$$

The ratio between initial and terminal threshold scales is at most $2h^2 n^3 2^w$. Its logarithm is $O(h \log(2nh))$, because $w = (h-1)\lceil \log_2(nh) \rceil$. Decreasing by $1 - 1/h$ gives the claimed number of levels. Finally $r_b \le R_b$ and $m \le n$ give the terminal sum bound. □

Example I.6 (A common large scale cancels)**.** Suppose every singleton capacity is 2^K, with $2^K \ge hn$, and consider a shift that retains this scale. There is one group and $t = \lfloor 2^K/(hn) \rfloor \ge 2^K/(2hn)$. Its residual-capacity upper bound satisfies $R < n 2^{K+1}$, so $R/t < 4hn^2$. The ratio of the initial threshold V_0/t to the stopping scale $V_0/(hnR)$ is therefore less than $4h^2 n^3$, independently of K. Thus this group's scan has $O(h \log(2nh))$ levels even if $K = \Theta(n)$. The baseline has used a small fraction of the available capacity to make later marginals small by diminishing returns. Random scale separation extends this cancellation to arbitrary unequal capacities; the width of a retained group introduces the extra factor h in Lemma I.5.

I.4 Poisson additivity makes a large increment analyzable

We now work with the residual objective g and its optimum G_*. This subsection supplies the smoothing and interpolation facts used by the algorithm. They hold for every normalized monotone DR objective, independently of how the capacity groups were constructed. Imagine splitting a count increment into many independent small arrival processes. For Poisson variables, their sum has exactly the same law as one arrival process with the summed mean:

$$Y \sim \mathrm{Pois}(a), \quad Z \sim \mathrm{Pois}(b), \quad Y \perp Z \quad \Longrightarrow \quad Y + Z \sim \mathrm{Pois}(a+b).$$

Thus a prescribed batch can be split or combined without changing its distribution. This is the property we need to relate a large count update to a submodular calculation on individual arrivals. The actual algorithm samples one count per type; the individual arrivals appear only in the proof.

Example I.7 (One million units, one Poisson count)**.** Suppose a direction adds 10^6 units of a type and its fractional step is $\delta = 10^{-4}$. A proof may attach an independent $\mathrm{Pois}(\delta)$ count to each unit. Their sum is $\mathrm{Pois}(100)$, which can be sampled directly. Adding an independent background $\mathrm{Pois}(z_e)$ gives $\mathrm{Pois}(z_e + 100)$. No million-unit expansion is needed in the algorithm. For the saturating objective $g(k) = \min\{1, k\}$, the gain from a zero background is $1 - e^{-100}$, not a million times the one-unit gain. Additivity preserves the distribution; diminishing returns still has to control the objective value.

Poisson smoothing. For $z \in \mathbb{R}^n_+$, let $Y(z)$ have independent coordinates $Y_e(z) \sim \mathrm{Pois}(z_e)$. Define

$$\Phi_g(z) = \mathbb{E}[g(Y(z))], \qquad h_z(q) = \mathbb{E}[g(Y(z)+q)], \quad q \in \mathbb{Z}^n_+. \tag{I.1}$$

These expectations are finite because $g(x) \le \sum_e x_e g(\mathbf{1}_e)$. Every fixed translation $q \mapsto g(y+q)$ is monotone DR-submodular; hence so is h_z. For fixed $C \in \mathbb{Z}^n_+$, a paired sample of a type-e marginal is

$$X_C = g(Y(z)+C+\mathbf{1}_e) - g(Y(z)+C), \qquad 0 \le X_C \le R(C,e) := \Delta_e g(C). \tag{I.2}$$

Its expectation is $\Delta_e h_z(C)$, and one sample uses two ordinary value queries. No algorithmic call to h_z or Φ_g is made.

The Poisson function is convenient for progress, but does not interpolate g at integer points: for $g(k) = \min\{1,k\}$, $\Phi_g(1) = 1 - e^{-1}$ whereas $g(1) = 1$. For rounding we use a second function, $F^{\mathrm{loc}}_g(z)$, obtained by independently rounding each coordinate to its floor or ceiling with its mean preserved. This is the unit-cube multilinear extension of Soma–Yoshida (Soma and Yoshida, 2018, Section 4.1). The next inequality is the bridge: once we prove that $\Phi_g(z)$ is large, the locally roundable value is at least as large. The approximation ratio will come from the progress analysis, rather than from this inequality alone.

Lemma I.8 (Local interpolation dominates Poisson smoothing)**.** *For every $z \ge 0$,*

$$F^{\mathrm{loc}}_g(z) \ge \Phi_g(z). \tag{I.3}$$

Proof. Fix all coordinates except e. The values of g as its integer eth coordinate increases form a concave sequence, since its successive differences are nonincreasing by DR. Its linear interpolation between consecutive integers is therefore a concave function. For any nonnegative integer random variable of mean z_e, Jensen's inequality bounds expected value by that interpolation at z_e. The floor–ceiling random variable of mean z_e attains the interpolation exactly.

Starting with the independent Poisson coordinates, replace coordinate one by this floor–ceiling variable, conditioning on every other coordinate. The expected objective does not decrease. Repeat for each coordinate with fresh independent floor–ceiling choices. At the end the expectation is $F^{\mathrm{loc}}_g(z)$. The linear growth bound on g ensures integrability throughout, despite the unbounded Poisson support. □

Lemma I.9 (Ray concavity and an existence consequence)**.** *For normalized monotone DR f, F^{loc}_f is monotone and concave on every nonnegative ray. In particular,*

$$F^{\mathrm{loc}}_f(cx) \ge cF^{\mathrm{loc}}_f(x) \quad (0 \le c \le 1).$$

For every rational x in an integral polymatroid Q,

$$\max_{X \in Q \cap \mathbb{Z}^n} f(X) \ge F^{\mathrm{loc}}_f(x).$$

Proof. Inside a unit cube, the partial derivative in coordinate e is an expectation of $\Delta_e f$ at rounded other coordinates. Coupling these roundings monotonically and using DR shows that this derivative is nonincreasing as any coordinate increases. At an integer boundary in coordinate e, its right derivative is at most its left derivative; other partial derivatives are continuous there. Along a nonnegative ray the directional derivative is therefore nonincreasing, including its one-sided values at cube boundaries. This proves concavity; all partial derivatives are nonnegative, proving monotonicity. Normalization gives the displayed scaling inequality. Finally apply Theorem I.25 at x for every $\xi > 0$. The finite integer optimum dominates the returned expectation; let ξ decrease to zero. This is an existence argument, not an additional algorithmic rounding call. □

The rounding theorem used in this existence argument is proved in Section I.9 for an arbitrary feasible rational point. Its proof is independent of the fractional construction, so this forward reference does not assume the approximation result being proved here.

I.5 Why one direction is split into a few colors

We want each fractional step to capture almost a δ fraction of the remaining optimum gap, using only a small number of count searches. Poisson additivity lets us implement a large count increment, but does not by itself prove this progress. There are two difficulties: individual unit gains can overlap heavily, and greedily choosing a whole bundle gives too weak a comparison with the optimum. Colors address both difficulties.

The two simple choices do not give what we need. One might score a direction by adding the marginal gains of its units at the current point. This can greatly overestimate a finite step. For $g(x) = \min\{1, x\}$ at $z = 0$, a direction of R units has initial directional derivative R, while its actual step gain is $1 - e^{-\delta R}$. Preserving the first-order prediction δR would require a step depending on R. This is precisely what we want to avoid when R may be enormous.

Instead, we could evaluate the whole direction as a single bundle, so overlap within it is already accounted for. But finding the best such bundle is again a submodular maximization problem. The inexpensive greedy construction available here can lose a constant fraction of the optimum gap. We need a direction-construction rule with almost the full gap, without solving that harder problem at every fractional step.

Several value contexts, one shared capacity. Maintain h count vectors $q_1, \dots, q_h$, called colors, whose sum v must be feasible. A color is just a label on a bundle of selected units; it is not a coordinate type or a separate feasible solution that we may use in addition to the others. The same type may occur in several colors. All colors share the constraint $\sum_j q_j \in Q_a$. There is no separate quota for a color: we do not give each color capacity r/h, and the colors need not be balanced or even all nonempty. Feasibility alone would indeed allow all selected units to be put in one color.

What makes colors behave differently is the *marginal-gain test*. At a positive threshold τ, an addition must both fit in the shared total $T = \sum_j q_j$ and have a sufficiently large marginal gain in its chosen color. The value test uses q_j and the common background $Y(z)$, without the other colors. Diminishing returns may make another unit unattractive in one color while the same unit is still valuable in an empty color. The routine tries every color, so it can use that opportunity. In the implementation, these are sampled threshold tests on the last unit of a proposed chunk; the following example has exact samples. Define

$$G_j = h_z(q_j) - \Phi_g(z), \qquad G = \sum_{j=1}^{h} G_j.$$

This lets different colors retain useful marginal gains that would disappear if every unit were evaluated against the entire union.

Example I.10 (Why two colors can accept more units than one)**.** Consider one type with $g(x) = \min\{1, x\}$ and shared constraint $Q_a = [0, 2]$. Start at $z = 0$ with two empty colors and threshold $\tau = 1$. Because $Y(0) = 0$, every marginal sample equals its true value.

1. One unit in color one has marginal $g(1) - g(0) = 1$, so it passes. A second unit in that color has marginal $g(2) - g(1) = 0$, so it fails although it would fit. The count search therefore returns one unit.

2. Color two is still empty. One unit there again has marginal $g(1) - g(0) = 1$ and fits the remaining shared capacity, so its search also returns one unit.

Thus the routine obtains $q_1 = q_2 = 1$ and $T = 2$. With only one color, the zero marginal of a second unit would leave the routine at $T = 1$ throughout all positive thresholds. Putting both units into color one would be feasible, but would not follow the algorithm's value test.

The two colors have total gain $G = 1 + 1 = 2$, while their union still has value $g(2) = 1$. Color gain measures progress through the finite-step lemma below; it is not the value of the union. For instance, a step of size δ in the two-unit direction has actual Poisson gain $1 - e^{-2\delta}$, compared with $1 - e^{-\delta}$ in the one-unit direction. This example illustrates one direction construction, not the final approximation ratio.

Nothing requires the routine to use every color. If sufficiently valuable additions fill the shared capacity in one color, the others may stay empty. The benefit of maintaining h colors is that there are h separate value contexts available when diminishing returns make such a concentration unhelpful. A color may contain many units; count searches and count-vector storage avoid processing those units individually.

Example I.11 (One greedy bundle can lose half the useful comparison)**.** There are three unit-capacity types a, b, c. We may take at most one of a, b, and at most one c. Type a covers a unit-weight feature x and a bonus feature of weight 0.1; b covers a different unit-weight feature y; c covers x. At $z = 0$,

a one-color greedy choice takes a, blocking b. Adding c now has zero marginal, so its value is 1.1, whereas the optimum $\{b, c\}$ has value 2.

With two colors, put a in the first. Type c still has marginal one in the empty second color and fits the shared capacity. The total color gain is now $1.1 + 1 = 2.1$, although the union still has value 1.1. There is no claim that this union is a better integral solution. It is a direction along which the algorithm takes a small fractional step, and the finite-step bound below converts the color gains into actual progress.

Here is why more colors improve the comparison. Temporarily ignore sampling errors, terminal threshold charges, threshold spacing, and geometric-search padding. The underlying greedy comparison in Section I.8 assigns the selected units a total threshold credit W, proves $G \geq W$, and bounds the optimum value missed by *each* color by that same W:

$$G_* - \Phi_g(z) - G_j \leq W.$$

Averaging over the h colors, then using $W \leq G$, gives

$$G_* - \Phi_g(z) \leq G + G/h, \qquad G \geq \frac{h}{h+1}\big(G_* - \Phi_g(z)\big).$$

For one color this reasoning supplies only half the gap. For $h = \Theta(1/\varepsilon)$ it supplies all but an $O(\varepsilon)$ fraction. The full sampled argument below proves the corresponding guarantee with explicit error terms. The point of the colors is the G/h term: the value already accumulated in one color is diluted when we average the comparison, while the gains of all colors are added.

Few colors also permit large fractional steps. We still have to turn G into a gain of Φ_g. For the proof, replace the independent arrivals within each color by a single coin that activates the whole color with probability $p = 1 - e^{-\delta}$. Submodular concavity makes this a lower bound on the true Poisson gain. Keep only events with exactly one active color. Each such event has probability

$$b = p(1-p)^{h-1} = (1 - e^{-\delta})e^{-\delta(h-1)}.$$

Consequently the step gains at least bG. Only the number of colors appears in this coefficient. Applying the same exactly-one argument to R separate units would instead give $p(1-p)^{R-1}$, which can be tiny.

Example I.12 (Why a whole-color event supplies a finite-step bound)**.** Suppose two colors each contain one token covering the same unit feature, with zero background. Independent Poisson arrivals retain the feature with probability $2p - p^2$. Keeping only exactly-one-color events gives $2p(1-p)$, a valid lower bound with slack p^2 from the omitted both-active event. The proof works the same way for large color bundles, after first replacing each bundle by a whole-color coin.

For scale, take a direction with one million units in 20 colors and $\delta = 1/400$. The color coefficient satisfies $b/\delta > 0.95$, regardless of the units per color. The corresponding exactly-one-unit coefficient divided by δ is less than e^{-2499}. Grouping keeps this particular finite-step bound useful without a million-unit step count.

We choose $h = \Theta(1/\varepsilon)$ and $\delta = 1/h^2$. Then $b = \delta(1 - O(1/h))$: the comparison loss and the finite-step loss are both small. The algorithm maintains only h color vectors and searches counts by type, so their number contributes an accuracy factor rather than a numerical-rank factor. There are only h^2 outer steps, and the residual point has denominator h^2. The final shrink changes this to h^3, which still depends only on accuracy and keeps local rounding fast. Thus colors balance two requirements: enough separate value contexts for the greedy comparison, but few enough groups for inexpensive steps and short rounding representations. The whole-color coins are only a proof device; the implemented sampler still draws Poisson counts by type.

Lemma I.13 (A finite-step inequality)**.** *For any $z \geq 0$, step $\delta > 0$, integer $h \geq 1$, and integer color vectors $q_1, \ldots, q_h \geq 0$, let $v = \sum_j q_j$ and put $p = 1 - e^{-\delta}$ and $b = p(1-p)^{h-1}$. Then*

$$\Phi_g(z + \delta v) - \Phi_g(z) \geq b \sum_{j=1}^{h} \big(h_z(q_j) - \Phi_g(z)\big). \tag{I.4}$$

Proof. **Step 1: realize the finite increment by independent arrivals.** Represent the v_e units of each type by distinct tokens for analysis. Give every token an independent $\mathrm{Pois}(\delta)$ arrival count. Summing within a type gives $\mathrm{Pois}(\delta v_e)$; adding an independent background $Y_e(z)$ gives $\mathrm{Pois}(z_e + \delta v_e)$. Different types remain independent. Thus the resulting value has expectation exactly $\Phi_g(z + \delta v)$. Keeping only the first arrival of each token can only lower value, by monotonicity. Its retained indicator is Bernoulli with probability $p = 1 - e^{-\delta}$, independently over tokens.

Step 2: replace one color by a whole-color coin. Condition on the background $Y(z) = y$ and on all retained tokens outside one color. Let $H(A)$ be the objective as a function of that color's retained token subset. By lattice DR, this is a monotone submodular set function. Write $\Psi(t)$ for its multilinear expectation when every token of that color is independently retained with probability t. Nonpositive mixed partials imply $\Psi''(t) \le 0$, so

$$\Psi(p) \ge (1-p)\Psi(0) + p\Psi(1).$$

The right side is the expectation when the whole color is activated with probability p, or no token of the color is activated otherwise. Replacing independent token indicators by this one independent color coin therefore lowers or preserves conditional expected value. Average over what was conditioned on and repeat for each color. The same argument is valid after earlier colors have been replaced: condition on their realized whole-color choices as part of the outside context.

Step 3: keep only patterns with one active color. At the end, the h whole-color coins are independent. For a fixed y, every activation pattern has gain at least zero over $g(y)$. The event that color j is the only active color has probability $b = p(1-p)^{h-1}$ and gain $g(y + q_j) - g(y)$. Keeping only these events gives expected gain at least $b\sum_j (g(y + q_j) - g(y))$. Averaging the background replaces these differences by $h_z(q_j) - \Phi_g(z)$. Combining the three successive lower bounds proves (I.4). The token and color replacements describe a coupling in the proof; the algorithm draws its Poisson backgrounds by type. □

I.6 Geometric search: keep the real direction feasible

The direction construction maintains h integer colors $q_1, \dots, q_h$ with real union $T = \sum_j q_j \in Q_a$. An update for $e \in B_b$ tests feasibility in its group with the *single original query*

$$a_b + T_b + k\mathbf{1}_e \in P,$$

where the query vector is zero outside B_b. All other groups remain feasible, so no full product-membership test is necessary. Values are always evaluated using the global objective $g(x) = f(a + x) - V$; objective interactions between groups are retained.

Fix a color q, threshold τ, and $\eta = 1/h$. Let $U = U_e - T_e$. Form the ordered set consisting of zero, the distinct integers $\lceil (1+\eta)^j \rceil \le U$ for $j \ge 0$, and $U + 1$. Search its indices by binary search. Zero is a vacuous passing endpoint; $U + 1$ is an unavailable upper endpoint. For an interior candidate k, reject on infeasibility; otherwise draw a fresh batch estimating

$$\Delta_e h_z(q + (k-1)\mathbf{1}_e),$$

and pass precisely when its empirical mean is at least τ. Terminate at adjacent grid indices and return the passing count k. Only these k real units are added to the color and union.

The search stores certificates only at its two current endpoints. It does not assume that independently resampled empirical scores are monotone. The endpoint inequality below supplies the replacement for an adjacent-integer certificate. With exact integer search, the rejected endpoint would be $k + 1$, and its marginal context would lie in the updated real color. A geometric search can reject much farther away. The shadow sequence introduced below supplies the missing domination without adding those extra units to the actual feasible direction.

Algorithm I.2. GeometricChunkSearch(z, T, q, e, τ, M)

Input: a feasible real union T, a color q, and the type's group B_b.

Let $v_0 < \cdots < v_N$ be the count grid, with $v_0 = 0$ and $v_N = U_e - T_e + 1$.
Set $i = 0$ (passing) and $j = N$ (unavailable).
while $j > i + 1$ **do**
  Set $m = \lfloor (i + j)/2 \rfloor$ and $k = v_m$.
  if $a_b + T_b + k\mathbf{1}_e \notin P$ (one original feasibility query) **then**
    Set $j = m$, recording a feasibility rejection.
  else
    Draw M fresh paired samples of $\Delta_e g(Y(z) + q + (k-1)\mathbf{1}_e)$.
    If the mean is at least τ, set $i = m$; otherwise set $j = m$.
    Retain the corresponding passing or value-rejection record.
  end if
end while
return $k = v_i$ and its upper-endpoint record at $\ell = v_j$.

Only the returned k is added to T and q. The upper record is used in the proof, and the algorithm never adds the analytical padding.

Lemma I.14 (Search endpoints)**.** *The real update is feasible. A positive returned count has an actual passing last-unit batch. Its upper endpoint ℓ has a rejection record or is unavailable, and*

$$\ell - 1 \le k + \lfloor \eta k \rfloor.$$

Every search uses $O(1 + \log h + \log\log(r + 2))$ feasibility calls and at most that many sample batches, even with nonmonotone empirical scores.

Proof. The two endpoint records are invariants of index binary search. The grid has $O(1+\eta^{-1}\log(U+2))$ entries. Consecutive distinct rounded powers $k < \ell$ satisfy $\ell - 1 < (1 + \eta)k$ by using the last power that rounds to k; the initial pair $0, 1$ and truncation at $U + 1$ obey the same weak integer inequality. Index search needs the logarithm of the grid size many tests. Finally $U_e < 2u_e \le 2r$. □

Explicit generation of this grid is permitted here because only oracle queries are counted. It is not claimed to take doubly logarithmic arithmetic time. As specified in Section A, dense arithmetic and count bit length are outside the oracle bound.

For analysis only, add another $\lfloor \eta k \rfloor$ tokens of the same type and color immediately after each real chunk of length k. Call the resulting sequence the shadow sequence and its final color counts $\bar{q}_j$. Then

$$q_j \le \bar{q}_j \le (1 + \eta)q_j.$$

Its union may be infeasible. It is not an algorithmic direction and will never be rounded.

The next standard augmentation argument justifies the token matroid used for shadow closure and for local rounding. Only the proof expands counts into tokens.

Lemma I.15 (Token matroid and count augmentation)**.** *For any finite collection of tokens labeled by types, declaring a token set independent when its type-count vector belongs to P defines a matroid. The lifted objective $A \mapsto f(c(A))$, where $c(A)$ is its count vector, is monotone submodular. Every maximal feasible integer count vector has coordinate sum r.*

Proof. Heredity is downward closure of P. For augmentation, let feasible integer counts x, y satisfy $|x| < |y|$. If no type with $y_e > x_e$ can augment x, then for each such e some constraint containing e is tight at x. Indeed, violation by $x + \mathbf{1}_e$, together with integrality of both $x(A)$ and $\rho(A)$, requires $x(A) = \rho(A)$.

Unions of tight sets are tight. If A, B are tight, then

$$x(A) + x(B) = x(A \cup B) + x(A \cap B) \le \rho(A \cup B) + \rho(A \cap B) \le \rho(A) + \rho(B) = x(A) + x(B).$$

Equality holds throughout. Therefore the union U of all the chosen blocking tight sets is tight. Outside U one has $y_e \le x_e$, while $y(U) \le \rho(U) = x(U)$, contradicting $|y| > |x|$. Thus some type with $y_e > x_e$ augments x. For token sets of these counts, there is an unused token of this type in the larger set, proving matroid augmentation. DR proves the lifted submodular inequality by comparing the marginal of a new token at two nested token sets.

Finally, integral-polymatroid integrality and greedy optimization (Edmonds, 1970) give a feasible integer base of total size r. The augmentation argument forces every maximal feasible count vector to have that size. Tokens are analytical objects; the algorithm never enumerates an expanded ground set of r units. □

Lemma I.16 (Shadow closure and shadow value)**.** *If a search ends with an unavailable or feasibility-rejected upper endpoint, the shadow prefix immediately after that search spans every unused token of that type in the residual token matroid. Also*

$$h_z(\bar{q}_j) - \Phi_g(z) \le (1+\eta)(h_z(q_j) - \Phi_g(z)).$$

Proof. Apply Lemma I.15 to Q_a. Tokens of a type are interchangeable, and a token set is independent exactly when its count vector belongs to Q_a.

Suppose a search begins at real union T' and has upper endpoint ℓ. The later shadow prefix contains at least the counts $T' + (\ell-1)\mathbf{1}_e$. Let t be the largest feasible augmentation in type e from T'. A feasibility rejection implies $t < \ell$; unavailability implies $t \le U_e - T'_e = \ell - 1$. The shadow prefix therefore contains an independent subset of counts $T' + t\mathbf{1}_e$ which spans the next and, by interchangeability, every unused token of type e.

For the value assertion, let $d = \bar{q}_j - q_j \le \eta q_j$. DR gives

$$h_z(q_j + d) - h_z(q_j) \le \sum_e d_e \Delta_e h_z(q_j) \le \eta \sum_e (q_j)_e \Delta_e h_z(q_j) \le \eta(h_z(q_j) - h_z(0)).$$

The last inequality telescopes the units of q_j at earlier contexts. Since $h_z(0) = \Phi_g(z)$, the claim follows. □

I.7 Aggregate sampling error without a union bound over rank

A feasible chunk can contain exponentially many units. Sampling errors must therefore be charged to value rather than to the number of units. Accepted chunks pay from their own telescoping gains, as proved below. For missed optimum units, the small ranges occur at an optimum prefix that the algorithm never knows. We couple these unqueried marginals to the queried samples, then concentrate their aggregate error directly. Requiring every unit to be individually accurate would unnecessarily restore $\log r$ in the sample size.

For intuition, if $g(k) = \min\{1, k\}$ and an optimum base has many copies of this one type, its successive prefix ranges are $(1, 0, \dots, 0)$. Their sum is one even though the singleton ranges sum to the base size. The fixed weights in the aggregate argument preserve this saving through adaptive searches and later matchings.

Condition on the random shift and all deterministic initialization. Fix an optimum integer base O for g on Q_a, independently of sampling. Fix an ordering of its tokens and put

$$R_i = \Delta_{e_i} g(O_{<i}), \qquad \sum_i R_i = G_*.$$

A batch at a history-measurable context (z, C, e) draws M independent Poisson backgrounds and averages

$$X_C = g(Y(z) + C + \mathbf{1}_e) - g(Y(z) + C), \qquad 0 \le X_C \le R(C, e) := \Delta_e g(C).$$

Each sample uses two original value queries. Define, without evaluating, the coupled variables for every optimum token of the same type:

$$Z_i = g(Y(z) + C + O_{<i} + \mathbf{1}_e) - g(Y(z) + C + O_{<i}).$$

They satisfy $0 \le Z_i \le R_i$ and $Z_i \le X_C$ samplewise. Let $\widehat{\mu}_b$ be the actual empirical mean and let $\widehat{\nu}_{i,b}, \nu_{i,b}$ be the latent empirical and population means at batch b.

Let $B \ge 1$ be a deterministic upper bound on the number of batches over all outer iterations, conditional on initialization. For $R_i > 0$ define

$$D_i = \max_{b:e_b=e_i} |\widehat{\nu}_{i,b} - \nu_{i,b}|/R_i,$$

with an empty maximum zero. Let D_Q be the largest normalized queried error, using zero for a zero range.

Lemma I.17 (Weighted aggregate concentration)**.** *For every $\beta > 0$,*

$$\Pr\left[D_Q > \beta \text{ or } \sum_i R_i D_i > \beta G_*\right] \le 4Be^{-2M\beta^2}.$$

Consequently, with

$$\beta = h^{-2}, \quad \zeta = h^{-1}, \quad M = \left\lceil \frac{\log(4B/\zeta)}{2\beta^2} \right\rceil,$$

both inequalities hold on one event $\mathcal{E}$ of probability at least $1-\zeta$, covering all adaptive batches and all subsequent matchings to them.

Proof. For fixed i, the signed normalized error $E_{i,b}$ at any eligible batch is the centered mean of M variables with range length one. Conditional Hoeffding's lemma gives

$$\mathbb{E}[e^{\lambda E_{i,b}} \mid \mathcal{F}_{b-1}] \le e^{\lambda^2/(8M)}.$$

At an ineligible or unexecuted index set $E_{i,b} = 0$. Taking expectations and using the exponential of a maximum bounded by a sum,

$$\mathbb{E}e^{\lambda D_i} \le \sum_{b=1}^{B} \mathbb{E}(e^{\lambda E_{i,b}} + e^{-\lambda E_{i,b}}) \le 2Be^{\lambda^2/(8M)} \quad (\lambda > 0).$$

If $G_* > 0$, the fixed weights $w_i = R_i/G_*$ sum to one. Pointwise convexity, with no independence assumption between the errors, gives

$$\mathbb{E}\exp\left(\lambda \sum_i w_i D_i\right) \le \sum_i w_i \mathbb{E}e^{\lambda D_i} \le 2Be^{\lambda^2/(8M)}.$$

Exponential Markov with $\lambda = 4M\beta$ bounds the aggregate tail by $2Be^{-2M\beta^2}$. If $G_* = 0$, all latent ranges and errors are zero, so this tail assertion is immediate. The queried-error tail has the identical bound by conditional Hoeffding and a union bound over B batches. Add the bounds. □

A rejected batch at threshold τ now gives

$$\nu_{i,b} \le \widehat{\nu}_{i,b} + R_i D_i \le \widehat{\mu}_b + R_i D_i < \tau + R_i D_i. \tag{I.5}$$

An individual D_i need not be at most β. What matters is that after any later matching selects one rejection for each optimum unit, the total error is at most $\sum_i R_i D_i \le \beta G_*$. Thus the sample size pays for the number of actual batches, without a union bound over the optimum units.

I.8 One direction with separate threshold grids

At fixed z, start all h colors at zero. Process each nonempty capacity group in a fixed order. Within group B_b, run the complete threshold grid $\tau_{b,0}, \ldots, \tau_{b,K_b}$; at each threshold visit every type in B_b and every color, applying the geometric search of Section I.6. All colors and their real union are retained when moving between groups. Return the real colors and their union T. Thus feasibility separates by group, while the value contexts continue to include selections from other groups.

Algorithm I.3. GroupedSampledSplit(z, M)

Input: the retained groups, baseline, and each group's threshold grid.

Initialize $T = q_1 = \cdots = q_h = 0$.
for each nonempty residual group B_b in the fixed order **do**
  for $s = 0, \ldots, K_b$ **do**
    for each type $e \in B_b$ and each color $j = 1, \ldots, h$ **do**
      Let k be the count returned by **GeometricChunkSearch**$(z, T, q_j, e, \tau_{b,s}, M)$.
      Update $q_j \leftarrow q_j + k\mathbf{1}_e$ and $T \leftarrow T + k\mathbf{1}_e$.
    end for
  end for
end for
return $(T, q_1, \ldots, q_h)$.

Each type participates only in the grid of its own group. Counts from earlier groups remain in the colors used for every later value query. This avoids an extra factor for the number of groups while preserving the global objective's diminishing-returns interactions.

Write

$$G = \sum_j (h_z(q_j) - \Phi_g(z)), \qquad W = \sum_{\text{real chunks } (k,b,s)} k\tau_{b,s}.$$

On $\mathcal{E}$, every passing last-unit certificate gives

$$G \geq W - \beta h G_*. \tag{I.6}$$

Indeed, for a chunk starting at q, its k true marginals are at least its last marginal, whose error is at most $\beta\Delta_e g(q + (k-1)\mathbf{1}_e)$. Multiplying this range by k costs at most $g(q + k\mathbf{1}_e) - g(q)$. These charges telescope on each color, and each real color is feasible in Q_a, hence has g-value at most G_*. Only real feasible colors enter this telescoping budget; shadow counts are used solely in the comparison below.

Lemma I.18 (Comparison with past rejections)**.** *On $\mathcal{E}$, every final shadow color satisfies*

$$G_* - h_z(\bar{q}_j) \leq \frac{(1+\eta)W}{1-\kappa} + \kappa V_0 + \beta G_*.$$

Proof. Use the residual token matroid independently in each group B_b, with disjoint optimum and shadow pools and sufficiently many copies of each type. The optimum restriction O_b is a base of rank r_b. Add r_b free dummy tokens and truncate this augmented group matroid to rank r_b.

Process that group's shadow tokens in chronological order. Maintain an independent set I spanning the processed shadow prefix and a base B' consisting of I and unmatched tokens of O_b, initially $I = \varnothing$, $B' = O_b$. Skip a shadow token $v \in \mathrm{cl}(I)$. For any other v, $I + v$ is independent; the fundamental circuit in $B' + v$ contains an optimum token $o \in B' \setminus I$. Replace o by v and match o to this shadow position. Immediately before this position, o is outside the closure of the preceding shadow prefix: $I + o$ is independent and I spans that prefix. At the end, append dummies to complete I to a base and finish the matching. Each optimum token is matched once; dependent shadow positions remain unmatched.

Fix a color j and an optimum token o_i whose partner has threshold index $s \geq 1$ within its group. It was outside the closure of the entire preceding threshold pass. By Lemma I.16, the search for its type and color in that earlier pass cannot have ended at an unavailable or feasibility-rejected upper endpoint. It therefore has a value rejection at $\tau_{b,s-1}$. If the search started at q' and had upper endpoint ℓ, its queried context $C = q' + (\ell - 1)\mathbf{1}_{e_i}$ is dominated by the final shadow color, by Lemma I.14. Hence DR and (I.5) give

$$\Delta_{e_i} h_z(\bar{q}_j + O_{<i}) < \tau_{b,s-1} + R_i D_i.$$

For a partner in its group's first threshold pass use the upper bound $g(\mathbf{1}_{e_i}) \leq \theta_b$. This singleton is feasible in Q_a because it belongs to O. For a dummy partner, the optimum token lies outside the closure of the whole real shadow sequence in that group; use its last threshold pass and τ_{b,K_b}.

Now telescope in the fixed global optimum order. Monotonicity gives $h_z(\bar{q}_j + O) \geq g(O) = G_*$. Each matched real shadow position is charged at most its threshold divided by $1 - \kappa$, and every position is used at most once. The sum of the weights of *all* shadow positions is at most $(1 + \eta)W$. The total dummy charge is at most $\sum_b r_b \tau_{b,K_b} \leq \kappa V_0$ by Lemma I.5. Finally the aggregate latent error is at most βG_*. No matching, token expansion, closure test, or latent query is executed. □

Lemma I.19 (Direction gain)**.** *For $\eta = \kappa = 1/h$ and $\beta = 1/h^2$, every split on $\mathcal{E}$ satisfies*

$$G \geq (1 - 4/h)G_* - \Phi_g(z) - V_0/h \geq G_* - \Phi_g(z) - 5A/h.$$

Proof. Average Lemma I.18 over colors and bound their average shadow gain by $(1 + \eta)G/h$ using Lemma I.16. Substitute $W \leq G + \beta h G_*$ from (I.6), and multiply by $1 - \kappa$:

$$(1 + \eta)\left(1 + \frac{1 - \kappa}{h}\right) G \geq \big((1 - \kappa)(1 - \beta) - (1 + \eta)\beta h\big)G_* \\ - (1 - \kappa)\Phi_g(z) - (1 - \kappa)\kappa V_0.$$

For $t = 1/h$, the G_* coefficient after division is

$$\frac{1 - 2t - 2t^2 + t^3}{1 + 2t - t^3} \geq 1 - 4t.$$

The numerator minus $(1 - 4t)$ times the denominator is $6t^2 + 2t^3 - 4t^4 \geq 0$. The magnitude of the coefficient of Φ_g is at most one, and that of V_0 at most t. Since $V_0 \leq A$ and $G_* \leq A$, the second inequality follows. □

I.9 Round only the remaining fractional units

We prove the local rounder for an arbitrary feasible rational point $z \in P$ and denominator D; its proof does not depend on the fractional algorithm. Here $F^{\mathrm{loc}} = F_f^{\mathrm{loc}}$ is the interpolation of the original objective f. In the final invocation, $z = (1 - 1/h)(a + z^L)$ and $D = h^3$.

Within this rounding subsection only, write $a = \lfloor z \rfloor$ for the local floor, independently of the earlier capacity baseline. The large integer part is already feasible. What remains is a yes-or-no decision for each coordinate with $v_e = z_e - a_e > 0$: round it up by one or leave it at the floor. There are at most n such decisions.

Example I.20 (Two rounding decisions despite a large numerical rank)**.** Let $P = \{x \geq 0 : x_1 + x_2 \leq 2{,}000{,}001\}$ and $z = (1{,}000{,}000.5, 1{,}000{,}000.5)$. After fixing the floor, at most one of the two coordinates can be rounded up. The local matroid is just the rank-one uniform matroid on two elements. Its two singleton bases average to $(1/2, 1/2)$, so the required base list has length two in this example. Rounding chooses which coordinate receives the last unit; none of the two million fixed units becomes a matroid element.

The permitted sets of round-ups form a matroid on at most n elements, as in Soma–Yoshida's local reduction (Soma and Yoshida, 2018, Section 4.2.2). Independently rounding each coordinate would realize $F^{\mathrm{loc}}(z)$, but could violate feasibility. The geometric matroid rounder instead makes the choices jointly and returns a feasible set. Its required input is an explicit list of bases, so we still have to turn the fractional vector v into such a list.

Ask for enough coverage rather than an exact representation. Soma–Yoshida obtain an exact convex decomposition using Cunningham's algorithm, with the stated $O(n^8)$ cost (Soma and Yoshida, 2018, Section 4.2.2). We use monotonicity to weaken the requested representation: find feasible bases whose average q satisfies

$$q \geq (1 - \alpha)v.$$

Every coordinate then receives almost its requested rounding frequency; additional coverage helps. The approximation concerns these frequencies, not feasibility of the bases. This is exactly the job of **CoverByBases**, whose returned list is passed directly to **IndependenceRound**.

Example I.21 (Coverage can be easier than equality)**.** Suppose the local bases are $\{1, 3\}$ and $\{2, 3\}$, and $v = (1/2, 1/2, 1/2)$. Their average is $q = (1/2, 1/2, 1)$, which covers v. Exact equality using bases is impossible: their averages have total mass two while v has total mass $3/2$. Monotonicity means that the extra half unit on coordinate three helps. We can therefore pass this base list directly to the independence rounder.

The small denominator guaranteed by the algorithm now matters. Each positive v_e is at least $1/D$, so the covering routine needs only $O(D\alpha^{-2}\log n)$ bases. With $D = h^3$ in our final call, this number depends on accuracy and logarithms, not on numerical rank. A rank-dependent step size in the fractional phase would have reintroduced a large denominator here, even though the local matroid has only n elements.

The next lemma supplies the local matroid and its oracle. We state its guarantee now and defer its proof until both rounding routines have been described, so the complete construction can be read first.

Lemma I.22 (The local matroid)**.** *For $a = \lfloor z \rfloor$ and $V_+ = \{e : z_e > a_e\}$, the family $\mathcal{I}_a = \{I \subseteq V_+ : a + \mathbf{1}_I \in P\}$ is a matroid, and $v = z - a \in P(M_a)$. One local independence query is one original integer-feasibility query. If Dz is integral, each positive v_e is at least $1/D$.*

LocalRound receives the fractional point, its denominator D, and an allowed value loss η. It fixes the floor, constructs the covering base list with loss allowance $\alpha = \eta/2$, and calls the previously proved geometric rounder with the other $\eta/2$. Its output is the floor plus the incidence vector of the returned local base.

Algorithm I.4. LocalRound$(P, z; D, \eta)$

$a \leftarrow \lfloor z \rfloor$; $v \leftarrow z - a$; $V_+ \leftarrow \{e : v_e > 0\}$.
if $V_+ = \varnothing$ **then**
  return a.
end if
Define the local matroid by $I \in \mathcal{I}_a$ exactly when $a + \mathbf{1}_I \in P$.
$\alpha \leftarrow \eta/2$.
$(B_1, \dots, B_t) \leftarrow$ **CoverByBases**(M_a, v, D, α).
$B \leftarrow$ **IndependenceRound**$((1/t, B_s)_{s=1}^t, \alpha)$.
return $a + \mathbf{1}_B$.

The two value losses are easy to keep separate. The covering average dominates the slightly shrunken target $(1-\alpha)v$, whose local multilinear value is at least $(1-\alpha)F^{\mathrm{loc}}(z)$. Rounding the base list preserves another $1-\alpha$ factor in expectation. Thus $(1-\alpha)^2 \geq 1-\eta$ gives the requested retention. The proof at the end of this subsection makes this calculation precise; the only new algorithmic task left is constructing the short covering list.

I.9.1 CoverByBases: prioritize coordinates with too little coverage

Suppose we have already chosen several bases. A coordinate that has appeared rarely relative to its target v_e should become more attractive in the next base. We implement this by maintaining positive weights, selecting a maximum-weight base, and decreasing the weights of the coordinates it covers. An uncovered coordinate keeps its weight and therefore gains relative priority. Division by v_e compares coverage with each coordinate's own request, rather than treating all frequencies as equal.

This is inexpensive because maximum-weight bases are found by ordinary matroid greedy (Proposition A.6), using at most n independence tests per iteration. The routine receives the local matroid, v, D, and α, and returns the actual bases it selected. There is no objective-value estimation in this stage.

Algorithm I.5. CoverByBases(M_0, v, D, α)

Restrict M_0 to $V_+ = \{e : v_e > 0\}$; set $m \leftarrow |V_+|$.
if $m = 0$ **then**
  return one empty base of the restriction.
end if
$\gamma \leftarrow \alpha/2$; $t \leftarrow \lceil 4D \log(2m)/\alpha^2 \rceil$; $w_e \leftarrow 1$ for every $e \in V_+$.
for $s = 1, \ldots, t$ **do**
  $p_e \leftarrow w_e / (\sum_{a \in V_+} w_a)$ for every $e \in V_+$.
  $B_s \leftarrow$ **MaximumWeightBase**$(M_0, (p_e/v_e)_{e \in V_+})$ (Proposition A.6).
  for $e \in V_+$ **do**
    $\ell_e \leftarrow \mathbf{1}[e \in B_s]/(Dv_e)$; $w_e \leftarrow w_e(1 - \gamma \ell_e)$.
  end for
end for
return $B_1, \ldots, B_t$.

Why must every coordinate eventually receive enough coverage? Each greedy base serves a definite fraction of the current total weight, so the total weight falls steadily. If one coordinate were badly undercovered, its own weight would remain too large to fit under that decreasing total. Comparing these two rates gives the frequency guarantee. The condition $v_e \geq 1/D$ keeps the multiplicative changes bounded and controls how many iterations the argument needs.

Example I.23 (Two iterations of the covering algorithm)**.** Let the local bases be $\{1, 3\}$ and $\{2, 3\}$, with $v = (1/2, 1/2, 1/2)$ and $D = 2$. Take $\alpha = 1/5$, so $\gamma = 1/10$. Initially all weights are one. If the first greedy tie is resolved in favor of $\{1, 3\}$, the update makes the weights $(0.9, 1, 0.9)$, because $1/(Dv_e) = 1$. The next maximum-weight base is $\{2, 3\}$, and the weights become $(0.9, 0.9, 0.81)$. Coordinate two was preferred because it had received no coverage; coordinate three is present in every base and its weight decreases at every iteration. The first two bases average to $(1/2, 1/2, 1) \geq v$. The algorithm continues for its prescribed t iterations; the potential proof guarantees coverage for the final average.

Lemma I.24 (Approximate covering by bases)**.** *For a matroid M_0 on N elements and $v \in P(M_0)$ with every positive coordinate at least $1/D$ and $0 < \alpha < 1/2$, Algorithm I.5 returns $t = O(D\alpha^{-2} \log(2N))$ bases of the positive-support restriction, whose average q satisfies $q \geq (1 - \alpha)v$ coordinatewise. It uses $O(Nt)$ independence queries and no objective-value queries.*

I.9.2 Justify the local reduction and the covering guarantee

The construction is complete. The first proof verifies that the remaining round-up decisions really form the promised small matroid. The second formalizes the weight comparison that makes the base list cover v. The final theorem combines these facts with the geometric rounder.

Proof of Lemma I.22. **Step 1: describe the residual feasible region.** Downward closure gives $a \in P$. Let $P_a = \{y \geq 0 : a + y \in P\}$. Define the integer-valued function

$$\rho_a(S) = \min_{U \supseteq S} (\rho(U) - a(U)).$$

All terms are nonnegative because a is feasible. The empty-set value is zero, and monotonicity follows because a larger S has fewer admissible supersets in the minimum. For minimizing supersets $U \supseteq S$ and $W \supseteq T$, submodularity of $\rho - a$ gives

$$\rho_a(S) + \rho_a(T) \geq (\rho - a)(U \cup W) + (\rho - a)(U \cap W) \geq \rho_a(S \cup T) + \rho_a(S \cap T).$$

Thus ρ_a is an integral polymatroid rank function. If $a + y \in P$ and $y \geq 0$, then for all $U \supseteq S$, $y(S) \leq y(U) \leq \rho(U) - a(U)$, giving $y \in P(\rho_a)$. Conversely, $\rho_a(S) \leq \rho(S) - a(S)$ shows that $y \in P(\rho_a)$ implies $a + y \in P$. Hence $P_a = P(\rho_a)$.

Step 2: impose the zero–one coordinate caps. Define

$$\sigma(S) = \min_{A \subseteq S} (\rho_a(A) + |S \setminus A|).$$

Every point in $P(\rho_a) \cap [0,1]^n$ satisfies the right-side upper bound for every A, and therefore lies in $P(\sigma)$. Conversely $\sigma(S) \le \rho_a(S)$ and $\sigma(\{e\}) \le 1$ show that $P(\sigma) \subseteq P(\rho_a) \cap [0,1]^n$. For submodularity, take minimizers $A \subseteq S$, $B \subseteq T$, apply submodularity of ρ_a, and use

$$|S \setminus A| + |T \setminus B| = |(S \cup T) \setminus (A \cup B)| + |(S \cap T) \setminus (A \cap B)|.$$

The union and intersection are admissible choices for the two resulting minima. Normalization and integrality are immediate. To check monotonicity for $S \subseteq T$, restrict a minimizing $B \subseteq T$ to $B \cap S$; both its rank term and its outside-cardinality term can only decrease. This proves $\sigma(S) \le \sigma(T)$.

Step 3: identify the matroid polytope and its oracle. The integer points of this capped integral polymatroid are zero–one vectors. Their supports satisfy augmentation by Lemma I.15 (or its tight-set proof), and therefore form a matroid. Edmonds' integrality theorem (Edmonds, 1970) identifies $P(\sigma)$ with their convex hull. Since $a + v = z \in P$ and $0 \le v < 1$, $v \in P(\sigma)$; restricting to its positive support gives $v \in P(M_a)$. This is the local rounding structure used in (Soma and Yoshida, 2018, Section 4). None of the analytical rank functions above is evaluated. The supplied feasibility oracle directly tests $a + \mathbf{1}_I \in P$, giving the required independence oracle. Finally $Dv_e = Dz_e - Da_e$ is a positive integer for $e \in V_+$, so $v_e \ge 1/D$. □

Proof of Lemma I.24. The empty support case is immediate. For the nonempty support, write w_e^s, p_e^s for the values at the start of iteration s, and put $a_e^s = \mathbf{1}[e \in B_s]/v_e$, so $\ell_e^s = a_e^s/D \in [0,1]$. The last bound follows from $v_e \ge 1/D$.

Step 1: each selected base covers enough weighted mass. The weights p_e^s/v_e are nonnegative. Since v is feasible, the maximum-weight independent set, and hence a maximum-weight base, has weight at least that of v. Therefore

$$\sum_e p_e^s a_e^s = \sum_{e \in B_s} p_e^s/v_e \ge \sum_e (p_e^s/v_e) v_e = 1, \qquad \sum_e p_e^s \ell_e^s \ge 1/D.$$

This is the only use of linear optimization in the potential analysis.

Step 2: upper-bound the total remaining weight. Let $W_s = \sum_e w_e^s$. The update gives $W_{s+1} = W_s(1 - \gamma \sum_e p_e^s \ell_e^s) \le W_s(1 - \gamma/D)$. Since $W_1 = m$ and $\log(1-u) \le -u$,

$$\log W_{t+1} \le \log m - \gamma t/D.$$

Step 3: lower-bound the weight of any one coordinate. For $0 \le u \le 1/2$, $\log(1-u) \ge -u - u^2$. Here $\gamma \ell_e^s \le \gamma < 1/2$, so for each e,

$$\log W_{t+1} \ge \log w_e^{t+1} = \sum_s \log(1 - \gamma \ell_e^s) \ge -\gamma(1+\gamma) \sum_s \ell_e^s,$$

using $(\ell_e^s)^2 \le \ell_e^s$. Combining the two potential bounds and multiplying by $D/(\gamma t)$ gives

$$\frac{1}{t} \sum_s a_e^s \ge \frac{1 - D \log m/(\gamma t)}{1+\gamma} \ge \frac{1 - \alpha/2}{1 + \alpha/2} \ge 1 - \alpha.$$

The middle inequality uses the specified t and $\gamma = \alpha/2$. Since $q_e = v_e t^{-1} \sum_s a_e^s$, this is the claimed coverage. Every iteration uses at most N independence queries for its greedy base, and the weight updates use no oracle. This proves the query bound. The method is multiplicative weights (Arora et al., 2012); the displayed calculation proves the particular covering statement needed here. □

Theorem I.25 (Denominator-aware approximate local rounder)**.** *For $z \in P$ with $Dz \in \mathbb{Z}^n$ and $0 < \eta < 1/2$, Algorithm I.4 returns an always-feasible integer X satisfying*

$$\mathbb{E} f(X) \ge (1 - \eta) F^{\mathrm{loc}}(z).$$

It uses no value queries and $O(nD^2\eta^{-5}\log^3(2n))$ expected integer-feasibility queries. The guarantee also holds for nonnegative, nonnormalized monotone DR objectives and for any integral polymatroid on at most n types.

Proof. If $v = 0$, the output is $a = z$ and the guarantee is exact. Otherwise Lemma I.22 supplies the local matroid and the positive-coordinate bound needed by Lemma I.24. Set $\alpha = \eta/2$ and let q be the returned base average, so $q \geq (1-\alpha)v$.

Define $g(I) = f(a+\mathbf{1}_I)$ on every subset of V_+. DR makes g monotone submodular; it is nonnegative but need not be normalized. Its multilinear extension $\mathcal{G}$ satisfies $\mathcal{G}(v) = F^{\mathrm{loc}}(z)$. Monotonicity and nonnegative-ray concavity give

$$\mathcal{G}(q) \geq \mathcal{G}((1-\alpha)v) \geq (1-\alpha)\mathcal{G}(v) + \alpha\mathcal{G}(0) \geq (1-\alpha)\mathcal{G}(v).$$

Theorem F.1 applies to the explicit base list and this nonnegative objective, returning a local base B with

$$\mathbb{E}g(B) \geq (1-\alpha)\mathcal{G}(q). \tag{I.7}$$

The algorithm needs no evaluation of g, because the rounder uses no value queries. By the definition of the local matroid, $X = a+\mathbf{1}_B$ is feasible on every run. Its expected value is at least $(1-\alpha)^2 F^{\mathrm{loc}}(z) \geq (1-\eta)F^{\mathrm{loc}}(z)$.

Finally the local rank is at most n and $t = O(D\eta^{-2}\log(2n))$. Covering uses $O(nt)$ feasibility queries; rounding uses $O(n(t-1)^2\eta^{-1}\log(2n))$ in expectation. Their sum is $O(nD^2\eta^{-5}\log^3(2n))$, with no factor of the numerical rank r. □

I.10 Approximation, feasibility, and original-oracle accounting

We now combine the construction's guarantees. Real directions are feasible independently of sampling accuracy, and final rounding is on the original constraint. The common sampling event is used only for the value analysis.

Approximation and feasibility in Theorem I.1. Fix a shift for now. Every real direction belongs to Q_a, so the final $z = L^{-1}\sum_i T^i$ belongs to Q_a and has denominator L. Thus $a + z \in Q$ and $y = (1-1/h)(a+z) \in P$ by Lemma I.3. Its denominator divides h^3. This feasibility statement is unconditional: it does not use $\mathcal{E}$. Theorem I.25 therefore gives integer feasibility on every run.

On $\mathcal{E}$, Lemma I.13 and Lemma I.19, with $\delta = 1/L$, imply

$$\Phi_g(z^{i+1}) \geq (1-b)\Phi_g(z^i) + b(G_* - 5A/h), \quad b = (1-e^{-\delta})e^{-\delta(h-1)}.$$

Starting from zero, write $c = 1-(1-b)^L$. Then $\Phi_g(z^L) \geq cG_* - 5cA/h$. Here $c \leq 1$ and

$$Lb \geq e^{-1/h} \geq 1 - 1/h, \qquad c \geq 1 - e^{-1} - 1/h.$$

The first inequality follows from $1 - e^{-\delta} \geq \delta e^{-\delta}$. Consequently

$$\Phi_g(z^L) \geq (1-1/e)G_* - 6A/h, \qquad V + \Phi_g(z^L) \geq (1-1/e-6/h)A.$$

By ray concavity, integer translation, and local domination,

$$\begin{aligned} F_f^{\mathrm{loc}}(y) &\geq (1-1/h)F_f^{\mathrm{loc}}(a+z^L) \\ &= (1-1/h)(V + F_g^{\mathrm{loc}}(z^L)) \\ &\geq (1-1/h)(V + \Phi_g(z^L)). \end{aligned}$$

Conditionally on every fractional trajectory, final rounding retains a further factor $1-1/h$ in expectation. The event $\mathcal{E}$ has conditional probability at least $1-1/h$; off the event, objective values remain nonnegative. Thus, writing $a_0 = 1-1/e$,

$$\mathbb{E}[f(X) \mid s] \geq (1-1/h)^3(a_0 - 6/h)A \geq (a_0 - 9/h)A.$$

All coefficients used here are positive for $h \geq 32$. Lemma I.4 gives

$$\mathbb{E}[f(X) \mid s] \geq (a_0 - 10/h)f(O|_H).$$

This also holds in either early-return case, since then $A = 0$ and Lemma I.4 forces $f(O|_H) = 0$. Finally average over the shift and use Lemma I.3:

$$\mathbb{E}f(X) \geq (a_0 - 10/h)(1-1/h)\mathrm{OPT} \geq (a_0 - 11/h)\mathrm{OPT} \geq (1-1/e-\varepsilon)\mathrm{OPT}.$$

The fixed optimum O in the last two displays is an original optimum; the optimum used for latent sampling and exchange was the independently fixed residual optimum for each shift. They have different roles. □

I.10.1 ORIGINAL-ORACLE ACCOUNTING

Let

$$L_n = \log(2nh), \qquad J = 1 + \log h + \log\log(r+2).$$

Up to absolute constants, J bounds the number of tests per geometric count search; this expression is positive and uniformly bounded below. There are $L = h^2$ outer steps, h colors, $O(h^2 L_n)$ thresholds for a type's group, and only n types in total. Thus a valid deterministic batch upper bound satisfies

$$B = O(nh^5 L_n J), \qquad M = O(h^4 \log(hB)).$$

For an explicit bound, let N_e be one plus the number of distinct positive grid counts at most U_e, and put $J_e = \lceil \log_2 N_e \rceil$. Every later search for type e has at most J_e tests because its residual interval can only shrink. Set

$$B = \max\left\{1,\ Lh \sum_b (K_b + 1) \sum_{e \in B_b} J_e \right\}.$$

This quantity is known before sampling and bounds every adaptive trajectory. It satisfies the displayed asymptotic estimate and does not require the unknown r.

Each feasibility test during direction construction changes just one group and costs one original feasibility call. Each sampled marginal costs two original value calls to f; the baseline constant cancels. Consequently

$$Q_f = O(nh^9 L_n J \log(hB) + n),$$

and the fractional stage has at most $O(nh^5 L_n J)$ original feasibility calls. Initialization contributes $O(n(1 + \log\log(r+2)))$ calls plus $O(n)$ value calls.

Final rounding is directly on the original P at denominator h^3. It costs

$$O\big(n(h^3)^2 h^5 \log^3(2n)\big) = O(nh^{11} \log^3(2n))$$

expected original feasibility calls and no value calls. Since $h = \Theta(1/\varepsilon)$, both L_n and J are $O(\Xi_\varepsilon)$, and $\log(hB) = O(\Xi_\varepsilon)$, for $\Xi_\varepsilon = \log(2n/\varepsilon) + \log\log(r+3)$. This proves the query bounds in Theorem I.1.

No hidden factor from the product constraint. Testing an arbitrary point in Q by testing every block could indeed cost m original queries. The algorithm never makes that request. During a split, feasibility of all unchanged blocks is an invariant, and a candidate modifies only one block. Final rounding uses P itself. The applications of local rounding to Q_a in the baseline preservation proof are existence arguments and incur no actual calls.

No hidden expansion in the scale range. Only occupied capacity groups are created. Their number is at most n, even if $\log r$ is enormous. The active interval width $w = O(h \log(nh))$ affects the logarithm of a group's threshold range, not the number of groups enumerated. Counts, powers, and grid entries may have large bit lengths. Their arithmetic and representation costs are not part of the theorem.

I.11 A LINEAR LOWER BOUND AT EXPONENTIAL RANK

Proposition I.26. *For any fixed approximation ratio $c \in (0, 1]$, every always-feasible randomized algorithm achieving $\mathbb{E}f(X) \geq c\mathrm{OPT}$ on all instances requires $\Omega(n)$ expected feasibility queries in the worst case, even for a known linear objective and modular polymatroid rank functions of rank exactly 2^n.*

Proof. Let $m = n - 1$, add a dummy coordinate d, and set $R = 2^n - 1$. For an unknown $j \in [m]$, use the box polymatroid with capacities one at coordinate j, zero at the other $m - 1$ ordinary coordinates, and R at d. Its rank function is

$$\rho_j(S) = \mathbf{1}[j \in S] + R\mathbf{1}[d \in S],$$

which is modular and has total rank 2^n. Give the objective explicitly as $f(x) = \sum_{i=1}^m x_i$. The optimum is one; a feasible integral output has value either zero or one, and obtaining one identifies j.

Value queries give no information about j. A feasibility query whose ordinary support contains at least two types, or whose positive ordinary count exceeds one, is infeasible regardless of j. Queries with no ordinary coordinate likewise give no information. The only informative queries are equality tests of a single candidate i: provided the dummy count is at most R, the answer is yes exactly when $i = j$.

Take j uniform on $[m]$. For any algorithm making at most t informative queries, the probability of identifying j is at most $(t + 1)/m$, even if an unverified final guess is allowed. Indeed, on the all-no path there are at most t tested indices plus one final guess; fix the random coins and then average. For an algorithm with a random query count N and success probability at least c, truncate after t queries. Markov's inequality gives

$$c \le \frac{t+1}{m} + \Pr(N > t) \le \frac{t+1}{m} + \frac{\mathbb{E}N}{t}.$$

Choose $t = \lfloor cm/2 \rfloor$. For sufficiently large m, $\mathbb{E}N = \Omega(c^2 m) = \Omega(n)$. Since this is an average over instances, some instance has at least that expected cost. □

Together with Theorem I.1, this proves a nearly linear oracle-complexity characterization in the exponential-rank regime. It does not depend on inferring an oracle lower bound from the number of bits in a capacity or in an output vector.

The upper bound uses approximate capacity exponents and never recovers all exact capacities. An information bound for exact capacity recovery would therefore concern a different task. Likewise, writing a large integer output can require many bits without forcing the same number of oracle calls. The upper and lower bounds resolve the dimension question in the stated oracle model, while leaving optimal accuracy, rank dependence, and arithmetic complexity open.