Dimension-Free Correlated Sampling for the Hypersimplex

Joseph (Seffi) Naor
Technion – Israel Institute of Technology, Haifa, Israel

Nitya Raju
University of Maryland, College Park, MD, USA

Abhishek Shetty
Massachusetts Institute of Technology, Cambridge, MA, USA

Aravind Srinivasan
University of Maryland, College Park, MD, USA

Renata Valieva
University of Maryland, College Park, MD, USA

David Wajc
Technion – Israel Institute of Technology, Haifa, Israel

Abstract

Sampling from multiple distributions so as to maximize overlap has been studied by statisticians since the 1950s. Since the 2000s, such correlated sampling from the probability simplex has been a powerful building block in disparate areas of theoretical computer science. We study a generalization of this problem to sampling sets from given vectors in the hypersimplex, i.e., outputting sets of size (at most) $k \in [n]$, while maximizing the overlap of the sampled sets. Specifically, the expected difference between two output sets should be at most α times their input vectors' ℓ_1 distance. A value of $\alpha = O(\log n)$ is known to be achievable, due to Chen et al. (ICALP'17). We improve this factor to $O(\log k)$, independent of the ambient dimension n. Our algorithm satisfies other desirable properties, including (up to a $\log^* n$ factor) input-sparsity sampling time, logarithmic parallel depth and dynamic update time, as well as preservation of submodular objectives. Anticipating broader use of correlated sampling algorithms for the hypersimplex, we present applications of our algorithm to online paging, offline approximation of metric multi-labeling, and swift multi-scenario submodular welfare approximating reallocation.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms

Keywords and phrases Correlated Rounding, Dependent Rounding

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.104

Related Version *Full Version*: `https://arxiv.org/abs/2511.13573`

Funding *Joseph (Seffi) Naor*: Supported in part by ISF grant 3001/24 and United States – Israel BSF grant 2022418.
Abhishek Shetty: Supported in part by ARO award W911NF-21-1-0328, the Simons Foundation, NSF award DMS-2031883, a DARPA AIQ award, an NSF FODSI Postdoctoral Fellowship and an Apple AI/ML Fellowship.
Aravind Srinivasan: Supported in part by NSF award number CCF-1918749.
Renata Valieva: Supported in part by NSF award numbers CCF-1918749 and CNS-2317194.
David Wajc: Supported in part by the Taub Family Foundation "Leader in Science and Technology" fellowship, Grand Technion Energy Program (GTEP) and ISF grant 3200/24.

1 Introduction

Consider the following natural *correlated sampling* problem: we wish to sample from numerous discrete distributions over $[n]$, or equivalently round numerous points in the n-dimensional probability simplex, $\Delta_n \triangleq \{\mathbf{x} \in [0,1]^n : \|\mathbf{x}\|_1 = 1\}$, in a *consistent manner*. In particular, for any two distributions with low total variation distance, i.e., points in Δ_n with low ℓ_1 distance, $\|\mathbf{x}-\mathbf{y}\|_1 \triangleq \sum_i |x_i - y_i|$, we wish the sets drawn according to these points to be similar in expectation.

This problem is well-understood [2,5,9,25,26,30]. Moreover, efficient correlated sampling algorithms have found numerous applications to disparate areas of computation, from approximation algorithms [9,25,30,37], parallel repetition [4,26], replicability and privacy [3,11], parallel sampling [1,32], cryptography [36], locality sensitive hashing [7,14], and more.

Such correlated sampling is a special case of what statisticians refer to as *coordinated sampling*, and have been studying since the 1950s [28]. Here, the motivation of maximizing overlap is in minimizing overhead of initiation interviews for newly sampled interviewees or overhead of hiring new interviewers for new regions with sampled interviewees. Such rationale was used and adopted, for example in the U.S. Bureau of the Census household surveys. See discussion of such applications and nearly half a century of work on such questions in the survey by Ernst [22].

In this work we study the following correlated sampling problem, introduced by [16], generalizing from the probability simplex to the convex hull of the hypersimplex and the origin, i.e., the polytope whose extreme points are vectors in $\{0,1\}^n$ with at most k ones,

$$\Delta_{n,k} \triangleq \{\mathbf{x} \in [0,1]^n : \|\mathbf{x}\|_1 \le k\}.$$

We wish to round points in this set while preserving marginals, respecting the cardinality constraint of k, and guaranteeing that the expected distance of the output sets for any two points $\mathbf{x}, \mathbf{y} \in \Delta_{n,k}$ is bounded in terms of the points' ℓ_1 distance. Formally, we study the following problem.

▶ **Definition 1.** *A distribution $\mathcal{D}$ over deterministic algorithms $\mathcal{A} : \Delta_{n,k} \to 2^{[n]}$ mapping points in $\Delta_{n,k}$ to subsets of $[n]$ of cardinality at most k is an* ***α-stretch correlated sampling algorithm for $\Delta_{n,k}$*** *if it satisfies the following properties for all $\mathbf{x}, \mathbf{y} \in \Delta_{n,k}$:*

(P1) ***(Cardinality)*** *$|\mathcal{A}(\mathbf{x})| \in \{\lfloor\|\mathbf{x}\|_1\rfloor, \lceil\|\mathbf{x}\|_1\rceil\}$ (hence $\le k$) for all $\mathcal{A}$ in the support of $\mathcal{D}$.*

(P2) ***(Marginals)*** *$\Pr_{\mathcal{A}\sim\mathcal{D}}[i \in \mathcal{A}(\mathbf{x})] = x_i$ for all $i \in [n]$.*

(P3) ***(Stretch)*** *$\mathbb{E}_{\mathcal{A}\sim\mathcal{D}}[\,|\mathcal{A}(\mathbf{x}) \oplus \mathcal{A}(\mathbf{y})|\,] \le \alpha \cdot \|\mathbf{x}-\mathbf{y}\|_1$.*

To clarify Property P3 and its use: after a common initialization step where we draw an algorithm $\mathcal{A} \sim \mathcal{D}$, we guarantee that when rounding any two vectors $\mathbf{x}, \mathbf{y} \in \Delta_{n,k}$, the output's expected distance (over the random choice of $\mathcal{A}$) is at most the vectors' ℓ_1 distance times some α. Naturally, we wish to keep α as small as possible.

For $k = 1$ it is known that α must be at least two [2,5],[1] and this bound is attainable, provided $\|x\|_1 = k = 1$, i.e., for $\Delta_n \subsetneq \Delta_{n,1}$ [2,9,25,26,30,32]. (As we show in the full version of the paper, a constant bound of $\alpha = 3$ for $\Delta_{n,1}$ more generally follows from the bound for Δ_n.) For $k > 1$, in contrast, [16] show that an $O(\log n)$ stretch is achievable, and use it for applications to metric multi-labelling, which we define later.

The stark difference between the case $k = 1$ and larger $k > 1$ is apparent. The natural question, which we study, is whether this "logarithmic in n" stretch is inherent to the problem.

[1] More precisely, some $\mathbf{x}, \mathbf{y}$ must have stretch at least $2/(1+\delta)$, for $\delta \triangleq \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_1$, though δ may tend to zero.

1.1 Our Contribution

Our main contribution is a correlated sampling algorithm for the hypersimplex with $k > 1$ with stretch logarithmic in k, but independent of the input dimension, n.

▶ **Theorem 2.** *There exists an $O(\log k)$-stretch correlated sampling algorithm for $\Delta_{n,k}$ for all n.*

Our algorithm is extremely fast, and can be implemented in $O(nnz \cdot \log^* n)$ time, for $nnz \leq n$ the number of nonzeros and $\log^* n$ the (extremely slowly-growing) iterated logarithm function (see Definition 3). Thus, the algorithm's running time is also *very nearly* linear in the input sparsity, and dimension independent. Our algorithm can further be implemented in $O(\log n \cdot \log^* n)$ parallel depth and $O(\log(nnz) \cdot \log^* n)$ update time in dynamic settings.

Property P2 immediately implies preservation of linear objectives. We also prove that our algorithm preserves *submodular* objectives. (See Section 3.4 for some background, but for now it suffices to say that these functions capture the ubiquitous phenomenon of diminishing returns.) In particular, we prove that our correlated sampling algorithm's output has at least as high a submodular value as independent rounding with marginals $\mathbf{x}$ (the so-called *multilinear extension*), while being oblivious to the objective submodular function, and also satisfying the cardinality constraint and desired low stretch guarantees. In the full version of the paper we extend this property to correlated sampling for partition matroids, where the ground set is partitioned into parts, with a cardinality constraint on each part.

Due to the widespread uses of correlated sampling for the probability simplex, we anticipate numerous subsequent applications of our correlated sampling algorithm for the hypersimplex. To illustrate this, we provide several simple applications of our algorithm, to online paging, offline metric multi-labeling, and dynamic submodular welfare maximizing reallocation. As these applications only illustrate the variety of (potential) applications of our correlated sampling for the hypersimplex, we defer their description to Section 4. However, for our last application, we note that our dimension-free stretch hints at further applications: combining it with a simple rounding algorithm for the case that k is large, we can obtain *constant stretch*, while approximately preserving marginals and submodular objectives. We believe this pattern will benefit future applications of our algorithm.

2 Preliminaries

Notation. Throughout, we use boldface letters $\mathbf{x}, \mathbf{y}$, etc., and use $\|\mathbf{x}\|_1 \triangleq \sum_i |x_i|$ to denote the ℓ_1 norm. In our running times, we use the standard iterated (binary) logarithm notation. (Throughout the paper, all logarithms are to the base two.)

▶ **Definition 3.** *For any n, the i-th iterated and the iterated logarithm are defined as follows.*

$$\log^{(i)} n \triangleq \underbrace{\log \log \ldots \log}_{i \text{ times}} n,$$

$$\log^* n \triangleq \min\{i \mid \log^{(i)} n \leq 1\}.$$

Sublinear sample time. Our algorithms' sampling time is sublinear in the input's dimension. Specifically, ignoring a modest $\log^* n$ factor, their running time is independent of n and only linear in the sparsity of the input vector $\mathbf{x}$, i.e., $nnz(\mathbf{x}) \triangleq |\{i \mid x_i \neq 0\}|$. (As $\mathbf{x}$ will be clear from context, we simply write nnz.) For this, we rely on a sparse representation that allows us to avoid reading the entire vector. In particular, we assume a natural representation allowing us to iterate through the non-zero coordinates of $\mathbf{x}$ in increasing order in time $nnz(\mathbf{x})$. Another operation we require is $O(1)$-time computation of standard bit-wise operations, XOR, AND, and NOT (see the full version).

Interface. Correlated sampling algorithms must use a common random seed for sampling, otherwise we may get different output sets in two different invocations of the sampling step for the same vector $\mathbf{x}$, violating Property P3 for vectors $\mathbf{x} = \mathbf{y}$. Fittingly, correlated sampling algorithms use a common *initialization* step, where we draw $\mathcal{A} \sim \mathcal{D}$, setting this randomness, and *sampling* steps, where we compute $\mathcal{A}(\mathbf{x})$. Note that in all our algorithms and the previous algorithms we make use of, the randomness of the initialization step can implicitly be "spread" over later sampling steps, drawing each random bit the first time it is needed. Therefore, all algorithms presented in this paper have constant initialization time, and so we avoid mentioning this, and only mention the sample time.

Previous Correlated Sampling Algorithms. For Δ_n, [2, 9, 25, 26, 30, 32] give 2-stretch correlated sampling algorithms. For $k > 1$, [16] provided an $O(\log n)$-stretch correlated sampling algorithm for $\Delta_{n,k}$, though a full proof of the latter is unavailable online.[2] Moreover, since we are interested in input-sparsity sample time and fast parallel and dynamic implementations of these, in the full version we provide full proofs of these algorithms, and implementations, with guarantees as in the following propositions. We also provide similar guarantees for $O(1)$-stretch correlated sampling in $\Delta_{n,1}$ more generally, by a simple reduction.

▶ **Proposition 4.** *There exists a 2-stretch $O(nnz)$ sample time correlated sampling algorithm for Δ_n. The algorithm can be implemented in $O(\log n)$ parallel depth and $O(n)$ work and using $O(\log(nnz)) = O(\log n)$ update time in dynamic settings.*

▶ **Proposition 5.** *There exists a correlated sampling algorithm for $\Delta_{n,k}$ for all $n, k \geq 1$ with sample time $O(nnz)$ and stretch $2\lceil \log n \rceil \leq 2 \log n + 2$. The algorithm can be implemented in $O(\log n)$ parallel depth and $O(n)$ work and using $O(\log n)$ update time in dynamic settings.*

Our input-sparsity implementation of [16] relies on constant-time lowest-common ancestor function, using which we only process relevant nodes in the postorder traversal of [16].

Bounding Stretch. Throughout, in our analysis we fix two vectors $\mathbf{x}, \mathbf{y} \in \Delta_{n,k}$, and prove that we satisfy all three desired properties, P1-P3. The following well-known observation [16] will simplify our discussion. As usual, $\mathbf{e}_i$ is the i^{th} basis vector in the standard basis for $\mathbb{R}^n$.

▶ **Observation 6.** *By triangle inequality, a correlated rounding algorithm has stretch α if and only if this stretch of α holds for any two points $\mathbf{x}$, $\mathbf{y} = \mathbf{x} + \varepsilon \cdot \mathbf{e}_i \in \Delta_{n,k}$, for infinitesimally small $\varepsilon > 0$.*

3 Composed Correlated Sampling

This section is dedicated to providing, analyzing, and implementing our main correlated sampling algorithm for the hypersimplex, whose main guarantee we restate here.

▶ **Theorem 2.** *There exists an $O(\log k)$-stretch correlated sampling algorithm for $\Delta_{n,k}$ for all n.*

We present our algorithm in Section 3.1, showing that it satisfies Properties P1 and P2, and analyze its stretch in Section 3.2. We discuss running times (in various computational models) in Section 3.3. We then prove that the algorithm preserves submodular objectives in Section 3.4.

[2] We thank Roy Schwartz for sharing a preprint of the full version of [16] with us.

3.1 The Composed Correlated Sampling Algorithm

We compose several correlated sampling algorithms and additional hashing and random thresholds. We precede its formal description and pseudocode (Algorithms 1–3) with a high-level layered description and motivation, interlaced with elements of its analysis.

3.1.1 Overview

Proposition 5 gives a correlated sampling algorithm for $\Delta_{n,k}$ with stretch $O(\log n)$. To obtain a smaller stretch, we aim, in a sense, to "project" the points onto a smaller dimension $d \ll n$, so as to obtain a stretch $O(\log d) \ll \Theta(\log n)$. Assuming the projection is in some sense invertible, this allows us to "lift" the same improved stretch back to the original (larger) dimension n.

A first attempt. Consider a vector $\mathbf{x}$. To try and achieve the desired invertible projection, we hash coordinates into $d \ll n$ buckets. In the pseudocode, $d = m^3$, where these parameters will be clarified below. Assuming (for now) that the sum of the x-values hashed to the same bucket is not greater than one, we give this bucket's coordinate a weight equal to the sum of the x-values hashed to it. Note that the obtained vector $\hat{\mathbf{x}} \in \mathbb{R}^d$ has the same ℓ_1 norm as $\mathbf{x}$. We now apply a correlated sampling algorithm to the vector $\hat{\mathbf{x}}$ and obtain a subset S of $[d]$ of size $\|\mathbf{x}\|_1$ (up to floor or ceiling). We then obtain a subset of $[n]$ with the right marginals, for each bucket b whose coordinate in $[d]$ is output in S, by picking a single coordinate i hashed to bucket b with probability $x_i / \sum_{j:h(j)=b} x_j$, using a simple correlated sampling algorithm for the case $k = 1$ (Proposition 4).

A second attempt. Unfortunately, the assumption that the sum of the x-values hashed to each bucket is at most one might be violated. However, by simple probabilistic arguments, one can show that taking sufficiently large $d \geq \text{poly}(k)$, then *small-valued* coordinates (whose individual x-values are at most, say, $1/10$) hashed to a bucket have sum exceeding one with probability at most $1/\text{poly}(d)$. We then hash *large-valued* coordinates, of which there are $O(k) \ll d$, into their own buckets, which are unlikely to collide with buckets to which other coordinates were hashed. Assuming no collisions or heavy buckets, the above is our desired invertible projection. In case of $1/\text{poly}(d)$-probability problematic events (heavy bucket or collisions), we fall back on some α-stretch algorithm. For now we think of $\alpha = O(\log n)$, though generally we can (and do) plug in other values of α. By linearity of expectation, this fall back algorithm contributes $\alpha/\text{poly}(d)$ to our algorithm's stretch, which is $O(1)$, provided $d \gg \alpha$. (This paragraph motivates our choice of $d = \text{poly}(m) = \text{poly}(k, \alpha)$.)

The full algorithm. Finally, our definition of small/large coordinates and heavy buckets can result in using two different correlated sampling algorithms (in different lines) to determine the output for different inputs $\mathbf{x}$ and $\mathbf{y}$ which only differ marginally in a single coordinate, i.e., $\mathbf{y} = \mathbf{x} + \varepsilon \cdot \mathbf{e}_i$. For example, this coordinate can be small for $\mathbf{x}$, but large for $\mathbf{y}$, or the change in this coordinate may result in one bucket being heavy when rounding $\mathbf{y}$, but not $\mathbf{x}$. This is extremely problematic, since this can result in our outputs for $\mathbf{x}$ and $\mathbf{y}$ being computed using different correlated sampling algorithms, and so these outputs are potentially uncorrelated (different) output sets, despite $\mathbf{x}$ and $\mathbf{y}$ being quite close. This results in stretch possibly as high as $O(k/\varepsilon)$. To overcome this issue, we randomize our thresholds for items being large or small, and for buckets being heavy. As we show, this results in the extremely problematic event outlined above happening with probability at most ε/k. Therefore, the

contribution to the stretch here is again only $O(k/\varepsilon) \cdot \varepsilon/k = O(1)$, and the final stretch is only $O(1) + O(\log d) = O(\log k + \log \alpha)$. Our final desired stretch of $O(\log k)$ follows by using this construction recursively, allowing us to plug in increasingly smaller values of α.

3.1.2 Algorithm Description

Our recursive algorithm relies on the correlated sampling algorithms of Propositions 4 and 5, and another correlated-sampling algorithm $\mathcal{A}$ for $\Delta_{n,k}$ with stretch α. This algorithm $\mathcal{A}$ can be the algorithm from Proposition 5, in which case $\alpha = 2\lceil \log n \rceil$, or (as we use in general) another correlated-sampling algorithm – with a smaller α – obtained by recursing this construction. (We later use this algorithm recursively so that in each level the stretch from $\mathcal{A}$ decreases and the overall stretch achieved is reduced.) At a high level, we rely on the algorithm for $k = 1$ with hashing to compress and map the input point $\mathbf{x} \in \Delta_{n,k}$ to a point in a lower-dimensional polytope $\hat{\mathbf{x}} \in \Delta_{m^3,k}$ for some $m^3 \leq n$. If we successfully lowered the dimension of the point to round in an invertible manner, we then round this point using the [16] correlated-sampling algorithm from Proposition 5 for this lower dimension m^3. When the compression does not succeed at some level of the recursion (due to hash collisions – this happens with small probability), we simply fall back to using algorithm $\mathcal{A}$ on the original n-dimensional input vector.

Our basic algorithm's pseudocode is given in Algorithms 1–3. On initialization (Algorithm 1) we initialize the randomness used later (in Algorithm 2) by the other correlated-sampling algorithms. In addition, we randomly hash the coordinates into some large $m \triangleq 2^5 k^5 \alpha^5$ many buckets. We then hash the coordinates and buckets into $[m^3]$, and draw two random thresholds $\sigma, \tau \sim \text{Uni}[0, 1/10]$. In Algorithm 2, we call a coordinate $i \in [n]$ *small* if $0 < x_i \leq 1/10 + \sigma$, else we call it *large*. If any two buckets or large coordinates are hashed to the same value in $[m^3]$ (a hash collision), or if the small coordinates in any one bucket have x-value summing to more than $1 - \tau$ (an unusually heavy bucket), we simply run Algorithm $\mathcal{A}$ on the input vector $\mathbf{x}$. Otherwise, we use Algorithm 3 to "compress" the small coordinates in each bucket (whose sum is less than one), using the $k = 1$ algorithm of Proposition 4: we obtain a single coordinate i called the *representative*, and have it "absorb" all the x-value of the small coordinates in its bucket – by taking on their x-values' sum and nullifying all small coordinates' x-values in its bucket (by resetting those to zero). Finally, using the (bijective) mapping from the remaining non-zero coordinates to $[m^3]$ (with representatives using their buckets' hash), we run the [16] algorithm on a vector $\hat{\mathbf{x}}$ of (smaller) dimension m^3. We then invert the bijective mapping from the remaining non-zero coordinates in $\mathbf{x}$ to $[m^3]$ to convert the output subset of $[m^3]$ to a subset of $[n]$, and return this set.

Algorithm 1 Initialization. ▷ Used before all runs of Algorithm 2

Set $m \leftarrow 2^5 k^5 \alpha^5$.
Initialize Algorithm $\mathcal{A}_1$ of Proposition 4 for $\Delta_{n,1}$. ▷ Init randomness
Initialize Algorithm $\mathcal{A}$ with a stretch of α for $\Delta_{n,k}$. ▷ Init randomness
Initialize Algorithm $\mathcal{A}_k$ of Proposition 5 for $\Delta_{m^3,k}$. ▷ Init randomness
for $b = 1, \ldots, m$ **do**
    $\mathcal{B}_b \leftarrow \emptyset$.
    Draw $R_b \sim \text{Uni}[m^3]$. ▷ Target hash for small items of bucket b
for $i = 1, \ldots, n$ **do**
    Draw $C_i \sim \text{Uni}[m^3]$. ▷ Target hash for coordinate i
    Add i to $\mathcal{B}_b$ for $b \sim \text{Uni}[m]$. ▷ Partition coordinates among buckets u.a.r.
Draw $\sigma \sim \text{Uni}[0, 1/10]$ and $\tau \sim \text{Uni}[0, 1/10]$.

Algorithm 2 Hypersimplex Correlated Sampling.

1: $S \leftarrow \{i \mid 0 < x_i \leq 1/10 + \sigma\}$. ▷ Non-zero small items' coordinates
2: $L \leftarrow \{i \mid x_i > 1/10 + \sigma\}$. ▷ Large items' coordinates
3: **if** $\exists h \in [m^3]$ such that $|\{i \in L \mid C_i = h\} \cup \{b \mid R_b = h\}| > 1$ **then** ▷ Hashing collision
4: **Return** $\mathcal{A}(\mathbf{x})$.
5: **if** $\exists b \in [m]$ such that $\sum_{i \in S \cap \mathcal{B}_b} x_i > 1 - \tau$ **then** ▷ Bucket too heavy
6: **Return** $\mathcal{A}(\mathbf{x})$.
7: $(\hat{\mathbf{x}}, \mathbf{REP}) \leftarrow \text{COMPRESS}(\mathbf{x}, S)$.
8: $T \leftarrow \mathcal{A}_k(\hat{\mathbf{x}})$.
9: **Return** $\{i \mid C_i \in T\} \cup \bigcup_{b:R_b \in T} REP_b$.

Algorithm 3 COMPRESS$(\mathbf{x}, S)$.

1: **for** $b = 1, \ldots, m$ **do**
2: $S_b \leftarrow S \cap \mathcal{B}_b$. ▷ Small items in bucket b
3: **if** $\sum_{i \in S_b} x_i > 0$ **then**
4: $REP_b \leftarrow \mathcal{A}_1\left(\frac{x_i}{\sum_{i \in S_b} x_i} \cdot \mathbb{1}[i \in S_b]\right)$. ▷ Single representative for small items in b
5: **for** $i \in S_b$ **do**
6: $x_i \leftarrow \mathbb{1}[i \in REP_b] \cdot \sum_{i \in S_b} x_i$. ▷ Compress small items
7: $\hat{\mathbf{x}} \leftarrow \mathbf{0}^{[m^3]}$.
8: **for** $i = 1, \ldots, n$ with $x_i > 0$ **do** ▷ Map non-zero coordinates to smaller vector
9: **if** $i \in REP_b$ for (unique) b **then**
10: $\hat{x}_{R_b} \leftarrow x_i$.
11: **else**
12: $\hat{x}_{C_i} \leftarrow x_i$.
13: **Return** $(\hat{\mathbf{x}}, \mathbf{REP})$.

3.1.3 First Observations

We note that Algorithm 2, which we denote henceforth by ALG, outputs a set of cardinality at most k, i.e., it satisfies Property P1.

▶ **Observation 7.** $|ALG(\mathbf{x})| \in \{\lfloor \|\mathbf{x}\|_1 \rfloor, \lceil \|\mathbf{x}\|_1 \rceil\}$ *for all* $\mathbf{x} \in \Delta_{n,k}$.

Proof. Follows by Property P1 of $\mathcal{A}$, whose output ALG may return in either Line 4 or 6, and by the same property of $\mathcal{A}_k$, since if ALG returns in Line 9, then since $\sum_i \hat{x}_i = \sum_i x_i$ and $|ALG(\mathbf{x})| = |\mathcal{A}_k(\hat{\mathbf{x}})|$ in this case, we have $|ALG(\mathbf{x})| = |\mathcal{A}_k(\hat{\mathbf{x}})| \in \{\lfloor \|\hat{\mathbf{x}}\|_1 \rfloor, \lceil \|\hat{\mathbf{x}}\|_1 \rceil\} = \{\lfloor \|\mathbf{x}\|_1 \rfloor, \lceil \|\mathbf{x}\|_1 \rceil\}$. ◀

Similarly, we note that the output of Algorithm 2 contains each element $i \in [n]$ with probability x_i, i.e., it satisfies Property P2. In fact, we prove a slightly stronger property. Specifically, for $\mathcal{R}$ the randomness used by Algorithm 1 in Lines 5–11, we show the following.

▶ **Observation 8.** *For any realization* r *of* $\mathcal{R}$*, element* $i \in [n]$ *and vector* $\mathbf{x}$

$$\mathbb{P}r[i \in ALG(\mathbf{x}) \mid \mathcal{R} = r] = x_i.$$

Consequently, by total probability, $\mathbb{P}r[i \in ALG(\mathbf{x})] = x_i$.

Proof. If $\mathcal{R} = r$ implies that ALG outputs a set in Lines 4 or 6, then $[ALG(\mathbf{x}) \mid \mathcal{R} = r] = \mathcal{A}(\mathbf{x})$, and so by Property P2 of $\mathcal{A}$,

$$\mathbb{P}r[i \in ALG(\mathbf{x}) \mid \mathcal{R} = r] = \mathbb{P}r[i \in \mathcal{A}(\mathbf{x})] = x_i.$$

If $\mathcal{R} = r$ implies that ALG outputs a set in Lines 9 and that $i \notin S$, then $\hat{x}_{C_i} = x_i$ and by Property P2 of $\mathcal{A}_k$,

$$\mathbb{P}r[i \in ALG(\mathbf{x}) \mid \mathcal{R} = r] = \mathbb{P}r[C_i \in \mathcal{A}_k(\hat{\mathbf{x}}) \mid \mathcal{R} = r] = \hat{x}_{c_i} = x_i.$$

Finally, if $\mathcal{R} = r$ implies that ALG outputs a set in Line 9 and that $i \in S$, then $i \in ALG(\mathbf{x})$ iff both $i \in REP_b$ and $R_b \in \mathcal{A}_k(\hat{\mathbf{x}})$ hold. By Property P2 of $\mathcal{A}_k$ and $\mathcal{A}_1$, and since $\hat{\mathbf{x}}_{R_b} = \sum_{i \in S_b} x_i$ independently of REP_b (and generally $\hat{\mathbf{x}}$ is independent of R_b), we have that

$$\begin{aligned}\mathbb{P}r[i \in ALG(\mathbf{x})] &= \mathbb{P}r[R_b \in \mathcal{A}_k(\hat{\mathbf{x}}) \mid \mathcal{R} = r] \cdot \mathbb{P}r[i \in REP_b \mid \mathcal{R} = r] \\ &= \hat{\mathbf{x}}_{R_b} \cdot \frac{x_i}{\sum_{i \in S_b} x_i} = x_i.\end{aligned}$$

◀

The crux of the analysis is in bounding the algorithm's stretch, i.e., quantifying Property P3.

3.2 Analyzing the Algorithm's Stretch

Analysis overview

When rounding two vectors $\mathbf{x}, \mathbf{y} \in \Delta_{n,k}$, we have three possible scenarios, named in accordance with their intuitively increasingly poor conditional stretch.

▶ **Definition 9.** *A pair of runs of ALG on vectors $\mathbf{x}, \mathbf{y} \in \Delta_{n,k}$ (with the same randomness) is*

1. **Good** ***(G)*** *if ALG outputs a set for both $\mathbf{x}$ and $\mathbf{y}$ in Line 9,*
2. **Bad** ***(B)*** *if ALG outputs a set for both $\mathbf{x}$ and $\mathbf{y}$ in either Lines 4 or 6, and*
3. **Tragic** ***(T)*** *if ALG outputs a set for $\mathbf{x}$ in Lines 4 or 6 and for $\mathbf{y}$ in Line 9, or vice versa.*

The following observation and subsequent lemma motivate the above terminology.

▶ **Observation 10.** *For vectors $\mathbf{x}, \mathbf{y} \in \Delta_{n,k}$, the sets $X \triangleq ALG(\mathbf{x})$ and $Y \triangleq ALG(\mathbf{y})$ satisfy*

$$\mathbb{E}[|X \oplus Y| \mid T] \leq 2k,$$
$$\mathbb{E}[|X \oplus Y| \mid B] \leq \alpha \cdot \|\mathbf{x} - \mathbf{y}\|_1.$$

Proof. The bound for the tragic event is trivial, since $|X|, |Y| \leq k$ by Observation 7, and thus by triangle inequality, $|X \oplus Y| \leq |X| + |Y| \leq 2k$ always. Next, conditioned on the bad event, we have $X = \mathcal{A}(\mathbf{x})$ and $Y = \mathcal{A}(\mathbf{y})$, so the bound follows by Property P3 of the α-stretch correlated sampling algorithm $\mathcal{A}$. ◀

Note that the conditional stretch for the tragic event is unbounded, since it holds for arbitrarily small $\|\mathbf{x} - \mathbf{y}\|_1 = \varepsilon$, as opposed to the bounded stretch conditioned on a bad pair of runs. This justifies our choice of a more positive term for the latter event. We now justify the choice of best term for the good event (G): we show that it contributes $O(\log m) = O(\log \alpha + \log k)$ to the stretch, which is $o(\alpha)$ unless $\mathcal{A}$ is already an algorithm with our target stretch of $O(\log k)$. As we show later in Lemma 14, we have that $\mathbb{P}r[G] = 1 - o(1)$, and so this also implies that good runs have the smallest expected stretch of all three types of runs.

▶ **Lemma 11.** *For vectors* $\mathbf{x}, \mathbf{y} \in \Delta_{n,k}$*, the sets* $X \triangleq ALG(\mathbf{x})$ *and* $Y \triangleq ALG(\mathbf{y})$ *satisfy*

$$E\left[|X \oplus Y| \mid G\right] \cdot \mathbb{P}r[G] = O(\log k + \log \alpha) \cdot \|\mathbf{x} - \mathbf{y}\|_1.$$

Proof (Sketch). By Proposition 5, the output sets have distance $O(\log m^3) \cdot \|\hat{\mathbf{x}} - \hat{\mathbf{y}}\|_1 = O(\log k + \log \alpha) \cdot \|\hat{\mathbf{x}} - \hat{\mathbf{y}}\|_1$ conditioned on the fact that runs for $\mathbf{x}$ and $\mathbf{y}$ are good. Therefore we wish to bound $\mathbb{E}[\|\hat{\mathbf{x}} - \hat{\mathbf{y}}\|_1 \mid G]$ in terms of $\|\mathbf{x} - \mathbf{y}\|_1 = \varepsilon$. In the case that x_i and y_i are both small, or are both large, $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ differ by a single coordinate, hence $\|\hat{\mathbf{x}} - \hat{\mathbf{y}}\|_1 = \|\mathbf{x} - \mathbf{y}\|_1 = \varepsilon$. However if y_i is large but x_i is small, then $\|\hat{\mathbf{x}} - \hat{\mathbf{y}}\|_1 = 2x_i + \varepsilon = O(1)$. As σ is randomly chosen, this last event happens with probability $O(\varepsilon) = O(\|\mathbf{x} - \mathbf{y}\|_1)$, which then implies our desired bound by total expectation. See full version of the paper for details. ◀

Given the preceding bounds on the expected contribution to the stretch of good runs, we wish to upper bound the probability of a pair of runs on vectors $\mathbf{x}$ and $\mathbf{y}$ being bad or tragic. To this end, we first upper bound the probability of Algorithm 2 running $\mathcal{A}$ on the original input vector, by proving that the probability of a hash collision is low, conditioned on any realization of σ.

▶ **Lemma 12.** *For any* $\mathbf{x} \in \Delta_{n,k}$ *and possible realization* s *of* σ*,*

$$\mathbb{P}r\left[\exists h \in [m^3] \text{ such that } |\{i \notin S \mid C_i = h, x_i > 0\} \cup \{b \mid R_b = h\}| > 1 \;\middle|\; \sigma = s\right] \leq \frac{1}{m}.$$

Proof. Since $\sum_i x_i \leq k$, the number of elements $i \in L$ is at most $\frac{k}{1/10+\sigma} \leq 10k$, while the number of buckets is m. The number of pairs of such is therefore at most $\binom{10k+m}{2}$. Since hashing is done uniformly, the probability of any pair colliding is $\frac{1}{m^3}$. By union bound over all pairs and using $m = 2^5 k^5 \alpha^5 \geq 10k/(\sqrt{2}-1)$, and so $10k + m \leq \sqrt{2}m$, the probability of a hash collision is at most

$$\binom{10k+m}{2} \cdot \frac{1}{m^3} \leq \frac{(10k+m)^2}{2m^3} \leq \frac{1}{m}.$$ ◀

Next, we show that under the same conditioning, the probability of any bucket b being heavier than $\frac{4}{5} \leq 1 - \tau$ is likewise small.

▶ **Lemma 13.** *For any* $\mathbf{x} \in \Delta_{n,k}$ *and possible realization* s *of* σ*,*

$$\mathbb{P}r\left[\exists b \in [m] \text{ such that } \sum_{i \in S \cap \mathcal{B}_b} x_i > \frac{4}{5} \;\middle|\; \sigma = s\right] \leq \frac{1}{m}.$$

Proof (Sketch). We divide small items into *tiny* items, having x value at most some $\lambda = O(1/\log m)$, and *little* items. For some $\mathcal{B}_b$ to have x value of small items exceeding $4/5$ requires either the sum of x-values of tiny items in $\mathcal{B}_b$ to exceed its expectation by some constant, or at least some constant number C of coarse items to belong to $\mathcal{B}_b$. Both events have probability polynomially small in m, the former by the choice of λ and Chernoff bounds, and the latter by union bound over the (few) C-tuples of the (somewhat few) little items. Union bounding over these events and over all $b \in [m]$ then gives the lemma. See the full version for details. ◀

Using the preceding lemmas, we can now upper bound the probabilities of the bad and tragic events.

▶ **Lemma 14.** *The probability that a pair of runs of Algorithm 2 on* $\mathbf{x}$ *and* $\mathbf{y}$ *is bad or tragic satisfies*

$$\mathbb{P}r[B] \leq \frac{2}{m},$$
$$\mathbb{P}r[T] \leq \frac{30\varepsilon}{m}.$$

Proof (Sketch). The first bound follows by the union bound and the previous two lemmas. In contrast, if a pair of runs is *tragic*, then Algorithm 2 terminates in Line 9 for one input, but terminates early (Lines 4 or 6) for the other, due to either a hash collision or a heavy bucket. All (three) cases causing such an event require one of the $1/m$-probability events from the previous lemmas to occur, and the random thresholds for an item being large or for a bucket being full to fall in a range of width at most $\varepsilon = \|\mathbf{x} - \mathbf{y}|_1$. But due to $\sigma, \tau \sim \mathrm{Uni}[0, 1/10]$, any one of these three bad cases then occurs with probability $10\varepsilon/m$. The upper bound on $\mathbb{P}r[T]$ then follows by the union bound. See the full version for details. ◀

With the above in place, we are now ready to bound the expected stretch of Algorithm 2.

▶ **Theorem 15.** *For all* $\mathbf{x}, \mathbf{y} \in \Delta_{n,k}$*, the sets* $X \triangleq ALG(\mathbf{x})$ *and* $Y \triangleq ALG(\mathbf{y})$ *satisfy*

$$\mathbb{E}\left[|X \oplus Y|\right] = O(\log k + \log \alpha) \cdot \|\mathbf{x} - \mathbf{y}\|_1.$$

Proof. By Observation 6, we can assume that $\|\mathbf{x} - \mathbf{y}\|_1 = \varepsilon$. By total expectation over the good, bad and tragic events, using Lemmas 11 and 14, we get:

$$\begin{aligned}
\mathbb{E}[|X \oplus Y|] &= \mathbb{E}[|X \oplus Y| \mid G] \cdot \mathbb{P}r[G] + \mathbb{E}[|X \oplus Y| \mid B] \cdot \mathbb{P}r[B] + \mathbb{E}[|X \oplus Y| \mid T] \cdot \mathbb{P}r[T] \\
&\leq O(\log k + \log \alpha) \cdot \|\mathbf{x} - \mathbf{y}\|_1 + \alpha \cdot \frac{2}{m} \cdot \|\mathbf{x} - \mathbf{y}\|_1 + \frac{2k}{\varepsilon} \cdot \frac{30\varepsilon}{m} \cdot \|\mathbf{x} - \mathbf{y}\|_1 \\
&\leq O(\log k + \log \alpha) \cdot \|\mathbf{x} - \mathbf{y}\|_1,
\end{aligned}$$

where the last inequality follows from our choice of m satisfying $m \geq 2\alpha$ and $m \geq 60k$. ◀

Taking Algorithm $\mathcal{A}$ in Algorithm 2 to be the $\alpha = O(\log n)$-stretch algorithm of Proposition 5, Theorem 15 immediately implies an $O(\log k + \log \log n)$-stretch correlated sampling algorithm. We improve on this bound and obtain an $O(\log k)$ stretch by recursing our construction. We start by describing the construction.

Our recursive construction. For the base case, $\mathcal{A}'_0$ is the algorithm of Proposition 5. For all $i \geq 1$ we let $\mathcal{A}'_i$ be Algorithm 2 with Algorithm $\mathcal{A}$ in Lines 4 and 6 chosen to be $\mathcal{A}'_{i-1}$.[3] We denote by α_i the stretch of algorithm $\mathcal{A}'_i$. By Proposition 5, $\alpha_0 \leq 2 \log n + 2$. On the other hand, by Theorem 15, for general i we have for some constant $C > 1$ the recurrence

$$\alpha_{i+1} \leq C \cdot (\log k + \log \alpha_i).$$

Using this recursive construction we obtain our claimed $O(\log k)$ stretch, as follows.

▶ **Theorem 2.** *There exists an* $O(\log k)$*-stretch correlated sampling algorithm for* $\Delta_{n,k}$ *for all* n*.*

[3] We can also consider running $\mathcal{A}'_{i-1}$ in Line 9, but we focus on the simpler recurrence.

Proof. Let $p \triangleq 3C \cdot \log k + 10C^2$, and note that $10C^2/\log(10C^2) \geq 3C$ for all $C > 1$. Consequently, since $x/\log x$ is increasing for $x \geq 10C^2 > e$, we have that if $\alpha_i \geq p \geq 10C^2$, then $\alpha_i/3 \geq C \log \alpha_i$. Moreover, if $\alpha_i \geq p$, then trivially $\alpha_i/3 \geq p/3 \geq C \cdot \log k$. Combining the above we have that if $\alpha_i \geq p$ then $2\alpha_i/3 \geq C \cdot (\log k + \log \alpha_i) \geq \alpha_{i+1}$. Thus, since $\alpha_0 = O(\log n)$, for $i \triangleq \log_{3/2}(\alpha_0) = O(\log \log n)$, Algorithm $\mathcal{A}'_i$ has stretch $\alpha_i \leq p = O(\log k)$. ◀

By the preceding proof, to get an $O(\log k)$ stretch we only need a recursion depth of $O(\log \log n)$. In the following section we refine this bound and consider other computational aspects of implementations of Algorithm 2 and our derived recursive correlated sampling algorithm.

3.3 Computational Considerations

In our preceding proof of Theorem 2, we only used the fact that the stretch decreases by a constant factor until it is below some $O(\log k)$ term. However, since the dependence on n in the stretch of each algorithm $\mathcal{A}'_i$ is asymptotically *logarithmic* in the stretch of the preceding algorithm $\mathcal{A}'_i$, we can show that the first iterations decrease the stretch significantly faster and so significantly fewer levels of recursion are necessary to attain stretch $O(\log k)$, which translates into speedups for our algorithm.

▶ **Lemma 16.** *The i-th algorithm $\mathcal{A}'_i$ for all $i \leq \log^* n - 2$ has stretch $\alpha_i = O(i \cdot \log k + \log^{(i)} n)$.*

The proof, which is a simple inductive argument, is deferred to the full version. We also show there that after $\log^* n$ levels of recurrence, giving a stretch of $O(\log k \cdot \log^* n)$, by the halving argument from our previous proof of Theorem 2, we then decrease the stretch by a further $\log^* n$ factor in only $O(\log \log^* n)$ more levels of recursion, yielding the following.

▶ **Theorem 17.** *For some $i = \log^* n + O(\log \log^* n) = O(\log^* n)$, Algorithm $\mathcal{A}'_i$ has stretch $O(\log k)$.*

▶ Remark 18. One can show that the more involved recurrence mentioned in Footnote 3 converges exponentially faster: it allows to double the number of iterated logarithms in each level, thus improving on Lemma 16, yielding a stretch $\alpha_i = O(i \cdot \log k + \log^{(2^i)} n)$ for $i = O(\log \log^* n)$, and thus this number of recursive calls (asymptotically) suffices for a stretch of $O(\log k)$. For simplicity of exposition, we omit the details.

In the remainder of this section we discuss the computational ramifications of Theorem 17 in various computational models, the most immediate one being the classic centralized setting. In all the following theorems, we note that

▶ **Theorem 19** (Near-Input-Sparsity Sample Time)**.** *There exists an $O(\log k)$-stretch correlated sampling algorithm for $\Delta_{n,k}$ with sample time $O(nnz \cdot \log^* n)$ in the worst case.*

Proof. By Theorem 17, we can focus on implementing $\mathcal{A}'_i$ for some $i = O(\log^* n)$. Compression in each of the $O(\log^* n)$ levels of the recurrence requires $O(nnz)$ time, since we spend time proportional to each bucket's vector, by Proposition 4. The final call to the rounding algorithm of Proposition 5 (for whichever dimension $d \leq n$ we end up using) takes a further $O(nnz)$ time. Finally, returning from the recurrence and lifting the output subset to a larger universe requires $O(nnz)$ time in each level, for a further $O(nnz \cdot \log^* n)$ time. ◀

Our recurrence depth together with implementations of the basic correlated sampling algorithms (Propositions 4 and 5) also implies that our algorithm can be (i) parallelized (at the cost of increasing the sample time to be slightly superlinear in n), with depth near logarithmic. Similarly, and that it can (ii) be dynamized with the near-logarithmic same update time. (See the full version).

▶ **Theorem 20** (Parallel Implementation). *There exists an $O(\log k)$-stretch correlated sampling algorithm for $\Delta_{n,k}$ with sampling time $O(n \cdot \log^* n)$ and depth $O(\log n \cdot \log^* n)$.*

▶ **Theorem 21** (Dynamic Implementation). *There exists an $O(\log k)$-stretch correlated sampling algorithm for $\Delta_{n,k}$ with update time $O(\log n \cdot \log^* n)$.*

3.4 Submodular Dominance

Since correlated sampling algorithms preserve marginals (Property P2), they naturally preserve linear objectives. In this section we show that our rounding scheme also preserves *subomdular* objectives that capture the ubiquitous notion of diminishing returns. We recall some basic definitions.

3.4.1 Submodularity: Technical Background

▶ **Definition 22.** *A set function $f : 2^E \to \mathbb{R}$ is* submodular *if it satisfies the* diminishing returns *property, namely*

$$f(e \mid S) \geq f(e \mid T) \qquad \forall S \subseteq T \subseteq E \setminus \{e\},$$

where $f(e \mid S) \triangleq f(S \cup \{e\}) - f(S)$ denotes the marginal value *(measured by f) of adding e to S.*

As we are interested in rounding-based algorithms, we need a way to extend (binary-valued) submodular functions to real vectors. We One natural such extension, corresponding to independent rounding, is the *multilinear extension* of [13].

▶ **Definition 23.** *The* multilinear extension *$F : [0,1]^E \to \mathbb{R}$ of a set function $f : 2^E \to \mathbb{R}$ is given by*

$$F(\mathbf{x}) \triangleq \sum_{S \subseteq E} f(S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i).$$

The maximum multilinear objective subject to any solvable polytope (i.e., a polytope over which one can optimize linear objectives efficiently) can be $(1 - 1/e)$-approximated in polynomial time [13]. This is optimal even subject to cardinality constraints, under standard complexity-theoretic assumptions [23] and information-theoretically given only value oracle access to f [33, 34].

Given the above approximability of the multilinear extension, it is natural to wish to round vectors $\mathbf{x}$ while preserving the given constraints, as well as preserving the multilinear objective. This is known to be achievable for matroid constraints and other constraints if f is known [13, 15]. An *objective-oblivious* counterpart to the above is given by the following definition.

▶ **Definition 24.** *A random vector $\mathbf{X} \in \{0,1\}^E$ satisfies* submodular dominance *if for any submodular function $f : 2^E \to \mathbb{R}$, we have that*

$$\mathbb{E}[f(\mathbf{X})] \geq F(\mathbb{E}[\mathbf{X}]).$$

It is known that tree-based pivotal sampling [38] (the core algorithm used by [16]) satisfies submodular dominance. This follows since tree-based pivotal sampling satisfies *negative association* (and more) [6, 12, 19, 21], and the latter is known to imply submodular dominance [17, 35]. We provide some relevant background on negative association here.

▶ **Definition 25.** *A random vector* $\mathbf{X}$ *is* negatively associated (NA) *if for any two increasing functions* f, g *of disjoint variables in* $\mathbf{X}$*, we have* $\mathrm{Cov}(f, g) \leq 0$.

A simple example of the above is given by the so-called *0-1 lemma* [20].

▶ **Proposition 26.** *Any random vector* $\mathbf{X} \in \{0,1\}^n$ *with* $\sum_i X_i \leq 1$ *is NA.*

It is well-known that NA is closed under products [27].

▶ **Proposition 27.** *If independent vectors* $\mathbf{X}$ *and* $\mathbf{Y}$ *are NA, then so is their concatenation,* $\mathbf{X} \circ \mathbf{Y}$.

A more involved example of NA is the output of [16]. This follows since the latter is the output of pivotal sampling [18, 38] (albeit with correlated random seed) with a tree order (see the full version), whose output is known to satisfy NA [6, 12, 31].

▶ **Proposition 28.** *The output of [16] is NA.*

Recall that our interest in NA is due to its implication of submodular dominance [17, 35].

▶ **Proposition 29.** *Let* f *be a submodular function and* $\mathbf{X}$ *an NA vector. Then,* $\mathbb{E}[f(\mathbf{X})] \geq F(\mathbb{E}[\mathbf{X}])$.

3.4.2 Submodular Dominance of Algorithm 2

In this section we show that the output of Algorithm 2 satisfies submodular dominance, using the connection of the latter to negative association.

▶ **Theorem 30.** *The output of Algorithm 2 with* $\mathcal{A}$ *and* $\mathcal{A}_k$ *satisfying submodular dominance itself satisfies submodular dominance.*

Proof. Fix a realization r of $\mathcal{R}$. This in particular fixes the definition of large and small items, the buckets and the mappings. If r is such that we return a set in Lines 4 or 6, then our output is $\mathcal{A}_k(\mathbf{x})$, which satisfies submodular dominance by Proposition 28, and so by Property P2 of $\mathcal{A}_k$,

$$\mathbb{E}[f(ALG(\mathbf{x})) \mid \mathcal{R} = r] = \mathbb{E}[f(\mathcal{A}_k(\mathbf{x})) \mid \mathcal{R} = r] \geq F(\mathbb{E}[\mathcal{A}_k(\mathbf{x}) \mid \mathcal{R} = r]) = F(\mathbf{x}).$$

Otherwise, given $T = \mathcal{A}(\widehat{\mathbf{x}})$, our Algorithm 2 outputs in Line 9 a set

$$S(T) \triangleq \{i \mid C_i \in T\} \cup \bigcup_{b:R_b \in T} Rep_b.$$

To argue submodular dominance, we define the following auxiliary submodular function:

$$\widehat{f}(T) \triangleq \sum_{\substack{(i_1,\ldots,i_r)\in(b_1,\ldots,b_r):\\ T\cap\bigcup_b\{R_b\}=\{R_{b_1},\ldots,R_{b_r}\}}} f\left((S(T)\cap L)\cup\bigcup_{j=1}^{r}\{i_j\}\right)\prod_{j=1}^{r}\frac{x_{i_j}}{\widehat{x}_{R_b}}.$$

Since the output of $\mathcal{A}$ is independent of Rep_b, for any realization of T we can imagine that (independently) for each $R_b \in T$, some Rep_b is drawn from $\mathcal{B}_b$ according to the discrete distribution $(x_i/\widehat{x}_{R_b})_{i\in\mathcal{B}_b}$, by Property P2 of $\mathcal{A}_0$. Thus $\widehat{f}(T) = \mathbb{E}[f(S(T))]$ according to this sampling of $\{Rep_b\}_b$. So, if we let $I \triangleq \{(i_1,\ldots,i_r) \in S^r \mid i_1,\ldots,i_r$ belong to distinct $\mathcal{B}_b\}$, then taking total probability over T, expanding $\widehat{f}(\cdot)$, and using the subomdular dominance of $\mathcal{A}$, we have:

$$\begin{aligned}\mathbb{E}[f(S(T))] &= \mathbb{E}[\widehat{f}(T)] \\ &\geq \sum_{T\subseteq[m^3]} \widehat{f}(T) \prod_{i\in T} x_i \prod_{i\in[m^3]\setminus T} (1-x_i) \\ &= \sum_{S\subseteq L} \sum_{(i_1,\ldots,i_r)\in I} f\left(L_s \cup \bigcup_{j=1}^{r}\{i_j\}\right) \prod_{i\in S} x_i \prod_{i\in L\setminus S} (1-x_i) \prod_{j=1}^{r} x_{i_j} \prod_{b:R_b\cap\bigcup_j\{i_j\}=\emptyset} (1-\widehat{x}_{R_b}).\end{aligned}$$

Fortunately, this unwieldy expression has a simple interpretation: this is precisely the submodular objective obtained by sampling each large item i independently with probability x_i, and sampling at most one element i per bucket $\mathcal{B}_b$ with probability x_i, independently of the large items and other buckets. But by Propositions 26 and 27, this distribution is NA, and since NA implies submodular dominance [17, 35], we obtain submodular dominance conditioned on $\mathcal{R} = r$. By Observation 8, this implies that $\mathbb{E}[f(ALG(\mathbf{x})) \mid \mathcal{R} = r] \geq F(ALG(\mathbf{x}))$. The submodular dominance for the unconditional distribution then follows since by total expectation over $\mathcal{R}$. ◀

4 Applications

Recall that correlated sampling algorithms for Δ_n have found numerous varied applications. In this section we discuss three applications of our rounding scheme: (i) an application to rounding fractional online paging algorithms, which yields worse algorithms than known, yet is simple and hints at more applications in online settings, (ii) an application to metric multi-labeling via a framework of [16], improving their approximation from $O(\log n)$ to $O(\log k)$, and (iii) a more complicated application to collusion-free and swift submodular welfare maximizing reallocation in dynamic settings. We conclude the latter with a discussion of combining dimension-free bounds with a simple rounding scheme to effectively obtain constant stretch, at the cost of only nearly achieving the marginals (Property P2), and hence incurring a $(1-\varepsilon)$ loss in submodular objectives.

We emphasize that our main message here is the variety of the applications, which we believe hints at potential future wide-ranging applications of our more general correlated sampling machinery.

4.1 Online Paging

Recall that in the (online) paging problem, a cache of size k is given, able to store any of n pages. At each time t a page $i_t \in [n]$ is requested. If i_t is not in the cache and the cache is full, this causes a *cache miss* and some other page must be *evicted* from the cache to make room for i_t. The objective is to minimize the number of evictions (hence cache misses), compared to the hindsight-optimal solution, measured in terms of the (multiplicative) *competitive ratio*.

A natural LP (linear programming) relaxation of the problem, with decision variables $y_{i,t} \in [0,1]$ corresponding to the extent to which page i is in the cache at time t, and $z_{i,t} = |y_{i,t} - y_{i,t-1}|$ the extent to which page i is evicted at time t, is as follows.

$$\begin{aligned}\min \sum_i \sum_t z_{i,t} & \\ \text{s.t. } y_{i_t,t} &\geq 1 && \forall t \\ y_{i,t} - y_{i,t-1} &\leq z_{i,t} && \forall i,t \\ y_{i,t} - y_{i,t-1} &\leq z_{i,t} && \forall i,t \\ y_{i,t} &\geq 0 && \forall i,t.\end{aligned}$$

No fractional online algorithm is better than $O(\log k)$-competitive with respect to this LP, and this ratio can be attained using the online primal-dual method [8]. We show that our correlated rounding algorithms provide $O(\log^2 k)$-competitive randomized integral algorithms. While better randomized algorithms with competitive ratio $O(\log k)$ have been known since the '90s [24], this hints at broader applicability of our correlated sampling algorithm for online problems.

Rounding fractional caches using correlated sampling. To round fractional caching algorithms we use our $O(\log k)$-stretch correlated sampling Algorithm 2 for $\mathcal{P}_{n,k}$, abbreviated by $\mathcal{A}$. For $\mathbf{x}^{(t)}$, the fractional cache at time t, our integral cache at time t is simply $\mathcal{A}(\mathbf{x}^{(t)})$. The random cache at each time t is feasible, since (i) $|\mathcal{A}(\mathbf{x}^{(t)})| \leq k$ by Property P1, and (ii) $\mathbb{P}r[i_t \in \mathcal{A}(\mathbf{x}^{(t)})] = y_{i,t}$ by Property P2, thus implying that this cache contains page i_t with probability one. On the other hand, the expected number of page misses at time t is, by Property P3, at most $|\mathcal{A}(\mathbf{x}^{(t)}) \oplus \mathcal{A}(\mathbf{x}^{(t-1)})| = O(\log k) \cdot \left\|\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)}\right\|_1 = O(\log k) \cdot \sum_{i,t} z_{i,t}$. Therefore, by linearity of expectation, the obtained (feasible) online paging algorithm incurs at most $O(\log k) \cdot \sum_{i,t} z_{i,t}$ evictions in expectation. Therefore, since the fractional online algorithm's objective $\sum_{i,t} z_{i,t} \leq O(\log k) \cdot OPT$, the randomized algorithm is at most $O(\log^2 k)$ competitive.

4.2 Metric Multi-Labeling

Chen et al. [16] introduced the correlated sampling problem which we study. Their motivation comes from multi-label classification problems, in which labels need to be assigned to objects, e.g., news articles, given some observed data. They considered the case where multiple labels can be assigned to objects, called the *metric multi-labeling* problem. It arises in various settings, e.g., classification of textual data such as web pages, semantic tagging of images and videos, and functional genomics. The assignment of labels to objects should be done in a manner that is most consistent with the observed data, from which two important ingredients are derived. The first is an *assignment cost* for every (object,label) pair, reflecting a recommendation given by a local learning process which infers label preferences of objects. The second is similarity information on pairs of objects, giving rise to *separation costs* incurred once different label sets are assigned to a pair of similar objects. The goal is to find a labeling that minimizes a global cost function, while taking into account both local and pairwise information.

Chen et al. [16] considered the setting in which the number of labels that an object can receive is at most $k \ll n$. They formulated the metric multi-labeling problem in this setting as a linear program that maps the m objects into a set of (fractional) vectors $\mathbf{y}^1, \ldots, \mathbf{y}^m \in \Delta_{n,k}$, where the i^{th} entry of vector j indicates the fraction of label i that object j receives. The objective function minimizes the global cost function of the (fractional) labeling, i.e., the sum of the assignment costs and the separation costs. The following lemma is proved in [16].

▶ **Proposition 31.** *If there is a polynomial-time α-stretch correlated sampling algorithm for $\Delta_{n,k}$, then the metric multi-labeling problem admits a polynomial-time α-approximation algorithm.*

The proof of the lemma is based on the following observations: (1) the preservation of the marginals in the correlated sampling algorithm guarantees that assignment costs are preserved in expectation; and (2) the α-stretch implies that separation costs are preserved in expectation with a loss of a factor α. As discussed in detail above and in the full version of the paper, [16] provided a polynomial-time $O(\log n)$-stretch algorithm, and from this they

obtain an $O(\log n)$-approximate algorithm for metric multi-labeling. Our main result that improves the stretch to $O(\log k)$ then implies the same improved approximation ratio for this problem. The latter is an asymptotic improvement of interest, as k is typically much smaller than n for this problem.

4.3 Swift and Collusion-Free Reallocation

We now turn to our most involved application. Part of our arguments are inspired by and build upon recent unpublished work of [10]. They studied the problem of low-recourse cardinality-constrained submodular maximization in a dynamic setting with unknown future.

In contrast, we show how our correlated sampling can be used to provide similar guarantees that are history-independent, and therefore collusion-resistant, given a sequence of potential scenarios that might require swift reallocation. This application relies on all properties of our algorithm, including submodular dominance (Theorem 30), and dimension-free stretch $O(\log k)$. We show that the latter property combines nicely with a simple constant-stretch algorithm which approximately preserves marginals and submodular objectives up to an additional $(1-\varepsilon)$ factor for the regime that $k \geq \text{poly}(1/\varepsilon)$ (by simple concentration arguments): this combination results in $O(\log(1/\varepsilon)$-stretch at the cost of a $(1-\varepsilon)$ loss in the objective, by running the appropriate (near-)correlated sampling algorithm depending on whether $k \geq \text{poly}(1/\varepsilon)$ or not. We turn to describing the problem we study in this section.

Submodular welfare maximization. In the basic submodular welfare (SWF) maximization problem, we have n buyers and m sellers. Each seller s sells some number k_s of homogeneous items from some universe U. Each buyer b in turn has a monotone submodular valuation function $f_b : U \to \mathbb{R}_+$ (i.e., $f_b(S) \leq f_b(T)$ for every $S \subseteq T \subseteq U$). The objective is to allocate these items, at most one of each, to the buyers, so as to maximize the social welfare, $\sum_b f_b(S_b)$, for $S_b \subseteq U$ the set of items allocated to b. We can encode the constraint that at most k_s items sold by s are allocated, at most one of which to each buyer, by requiring the (possibly fractional) allocation vector $\mathbf{x}$ with (s, b) to be a concatenation of m points in hypersimplexes, Δ_{n,k_s} with appropriate k_s.

$$\sum_b x_{b,s} \leq k_s \qquad \forall s$$
$$0 \leq x_{b,s} \leq 1 \qquad \forall b, s.$$

SWF is NP-hard to approximate within a factor of $(1 - 1/e)$ [29]. There are many approaches to attain this approximation factor in polynomial time. Most relevant to our subsequent dynamic SWF reallocation problem is the relax-and-round approach, as follows: the continuous greedy algorithm can be used to provide a $(1 - 1/e)$-approximation to the maximum multilinear extension subject to any solvable LP constraints, such as the above linear constraints [13]. One can then round the vectors using Algorithm 2, whose output satisfies the hypersimplex constraints (Property P1, matches the marginals (Property P2) and satisfies submodular dominance (see Theorem 30) (see Proposition 28), applying this algorithm to each sub-vector $\mathbf{x}^s$ with coordinates $\{(s, b) \mid b \in [n]\}$, corresponding to the cardinality constraint of k_s for items sold by s. By the algorithm's properties it satisfies all constraints, and as shown in the full version of the paper, this also satisfies submodular dominance for the entire vector (with the target marginals), and so the obtained value is at least as high as the multilinear extension, which gives us a $(1 - 1/e)$-approximation [13].

The dynamic problem. Suppose now that we have r possible *scenarios*, corresponding to some buyers or sellers leaving or entering the market. Let $SWF(i)$ denote the optimal SWF of scenario i. We wish to provide a good approximation of $SWF(i)$, while guaranteeing swift reallocation when switching from one scenario to the other. In particular, we wish to provide a β approximation (to be chosen shortly) for each configuration, while minimizing the maximum number of items reallocated. This can be captured by the following LP constraints, where $x_{b,s,i}$ is the fractional allocation to buyer b of items of seller s under scenario i:

$$\begin{aligned}
&\min R && && \text{(MMD-LP)}\\
&\text{s.t.} \sum_{b,s} z_{b,s,i,j} \le R && \forall i,j\\
&\quad x_{b,s,i} - x_{b,s,j} \le z_{b,s,i,j} && \forall b,s,i,j\\
&\quad x_{b,s,j} - x_{b,s,i} \le z_{b,s,i,j} && \forall b,s,i,j\\
&\quad \sum_{b} x_{b,s} \le k_s && \forall s \in \text{ scenario } i\\
&\quad \sum_{b} x_{b,s} \le 0 && \forall s \notin \text{ scenario } i\\
&\quad \mathbf{x} \ge \mathbf{0}.
\end{aligned}$$

The above formulation is still missing the SWF-approximation. Building on the approximate-or-separate method of [10] (as outlined in the full version), we can provide in polynomial-time a solution to the above constraint subject to the multilinear extension of the objective submodular function over $\mathbf{x}_{|i} \triangleq (x_{b,s,i})_{b,s}$ (the fractional assignment for scenario i) being a $(1-1/e-\varepsilon)^2$-approximation of $SWF(i)$ for each i. The question remains: how do we round all these vectors in a way to obtain bona fide *integral* solutions with low movement cost? [10] provide an approach that results in $O(R)$ movement cost (i.e., a constant-approximation of the optimal movement cost). Unfortunately, their algorithm is history dependent, which may incentivize buyers and sellers to time their arrivals and departures. Our correlated sampling algorithm, in contrast, is *history-independent*. Using our correlated sampling algorithm, which satisfies the hard cardinality constraints (Property P1), the target marginals (Property P2), $O(\log k)$ stretch (Theorem 2) and submodular dominance (Theorem 30), we obtain an algorithm with $(1-1/e-\varepsilon)^2$-approximation, using movement cost $O(R \cdot \log k)$, i.e., $O(\log k)$ times that of an optimal algorithm.

4.4 Discussion: From Dimension-Free to Constant

The above stretch for collusion-free reallocation can in fact be decreased to a *constant* for any constant $\varepsilon > 0$, and specifically to $O(\log(1/\varepsilon))$, albeit at the cost of a (negligible) $(1-\varepsilon)$ deterioration in approximation, and losing the history independence. Since this is subsumed by the history-dependent $O(1)$-stretch rounding of [10], we only outline the idea here briefly.

By sampling a uniform threshold for each coordinate, we can decide which elements to take to our solution (i.e., which item to sell to which buyer), using stretch $O(1)$: when $x_i \ge \tau_i$ for i.i.d., $\tau_i \sim \text{Uni}[0,1]$, we add i to the output. This yields the correct marginals independently, and so yields a set of expected value exactly equal to the multilinear extension, $F(\mathbf{x})$, and moreover results in a set of expected size $\|\mathbf{x}\| \le k$. To make this a feasible set, we can scale down the value by a factor of $1+\Theta(\sqrt{k})$, thus obtaining a set of expected size $k - \Theta(\sqrt{k \ln k})$. By simple concentration arguments, the probability that this set exceeds size k is polynomially small in k, and by submodularity, one can show that taking only

a subset of the sampled elements if more than k are sampled will only incur an expected additive loss of $(1/\text{poly}(k)) \cdot f(OPT)$. Combined with the $(1 - \Theta(\sqrt{k \ln k}))$ multiplicative loss from scaling down, this is a $1 - \varepsilon$ multiplicative factor and $\varepsilon \cdot f(OPT)$ additive factor, provided $k = \Omega(\log(1/\varepsilon)1/\varepsilon^2)$ is sufficiently large. On the other hand, by only keeping a subset of cardinality $\min\{k, |S|\}$ of the sampled elements S, it is easy to obtain stretch $O(1)$. Thus, we get a $(1 - \varepsilon)$ loss in objective (provided the target value is $\Omega(f(OPT))$) with $O(1)$ stretch if k is larger than some $\text{poly}(1/\varepsilon)$. Alternatively, if k is smaller than $\text{poly}(1/\varepsilon)$, then by applying this paper's main correlated sampling algorithm, we get no loss in the objective value, but stretch $O(\log k) = O(\log(1/\varepsilon))$, i.e., constant for any constant $\varepsilon > 0$. In either case, we incur a stretch no worse than a $1 - \varepsilon$ deterioration in approximation quality, and a constant stretch of $O(\log(1/\varepsilon))$. We believe this pattern will find future applications.

5 Summary and Open Questions

In this work we revisit the correlated sampling question for the hypersimplex, $\Delta_{n,k}$, and provide dimension-free stretch guarantees for the latter. We provide stretch $O(\log k)$ using a recursive algorithm of depth $O(\log^* n)$.

We note that any dimension-dependent stretch $f(n) \ll \log n$ should imply by similar approaches an $O(f(k))$ stretch with $O(f^*(n))$ levels of recurrence, where $f^*(n) \triangleq \min\{i \mid f^{(i)}(n) \leq 1\}$ is "the iterated f", defined using the i^{th} iteration of f, namely $f^{(i)}(x) \triangleq f(f^{(i-1)}(x)) \cdot \mathbb{1}[i > 0] + x \cdot \mathbb{1}[i = 0]$. However, better than logarithmic dependence on n is as yet unknown. Is this inherent, and is $\Theta(\log k)$ the optimal stretch given Properties P1 and P2? We leave this as a tantalizing open question.

We presented a number of applications, to online paging, metric multi-labeling and swift and history-independent dynamic reallocation for approximate submodular welfare maximization. While we do not see any of these applications as particularly compelling on their own, the variety of applications hints at further applications of our correlated sampling machinery. In particular, we discussed in Section 4.4, the dimension-independent stretch can, for some applications, be combined with simple rounding schemes to obtain *constant* stretch (while only approximately preserving marginals). We are optimistic that similarly to correlated sampling for the probability simplex, such (dimension-free) correlated sampling for the hypersimplex will find broader applications. We leave the search for such applications as a direction for future research.

References

1. Nima Anari, Ruiquan Gao, and Aviad Rubinstein. Parallel sampling via counting. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 537–548, 2024. doi:10.1145/3618260.3649744.
2. Omer Angel and Yinon Spinka. Pairwise optimal coupling of multiple random variables. *arXiv preprint arXiv:1903.00632*, 2019.
3. Ghazi Badih, Kumar Ravi, and Manurangsi Pasin. User-level private learning via correlated sampling. *Advances in Neural Information Processing Systems*, 2021.
4. Boaz Barak, Moritz Hardt, Ishay Haviv, Anup Rao, Oded Regev, and David Steurer. Rounding parallel repetitions of unique games. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 374–383, 2008. doi:10.1109/FOCS.2008.55.
5. Mohammad Bavarian, Badih Ghazi, Elad Haramaty, Pritish Kamath, Ronald L Rivest, and Madhu Sudan. Optimality of correlated sampling strategies. *Theory of Computing*, 16(1), 2020. doi:10.4086/TOC.2020.V016A012.

6 Petter Brändén and Johan Jonasson. Negative dependence in sampling. *Scandinavian Journal of Statistics*, 39(4):830–838, 2012.

7 Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29, 1997. doi:10.1109/SEQUEN.1997.666900.

8 Niv Buchbinder, Kamal Jain, and Joseph (Seffi) Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 15th Annual European Symposium on Algorithms (ESA)*, pages 253–264. 2007.

9 Niv Buchbinder, Joseph (Seffi) Naor, and Roy Schwartz. Simplex partitioning via exponential clocks and the multiway-cut problem. *SIAM Journal on Computing (SICOMP)*, 47(4):1463–1482, 2018. doi:10.1137/15M1045521.

10 Niv Buchbinder, Joseph (Seffi) Naor, David Wajc, et al. Chasing submodular objectives, and submodular maximization via cutting planes. *arXiv preprint arXiv:2511.13605*, 2025.

11 Mark Bun, Marco Gaboardi, Max Hopkins, Russell Impagliazzo, Rex Lei, Toniann Pitassi, Satchit Sivakumar, and Jessica Sorrell. Stability is stable: Connections between replicability, privacy, and adaptive generalization. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 520–527, 2023. doi:10.1145/3564246.3585246.

12 Jarosław Byrka, Piotr Skowron, and Krzysztof Sornat. Proportional approval voting, harmonic k-median, and negative association. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*, page 26, 2018.

13 Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing (SICOMP)*, 40(6):1740–1766, 2011. doi:10.1137/080733991.

14 Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002. doi:10.1145/509907.509965.

15 Chandra Chekuri, Jan Vondrak, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proceedings of the 51st Symposium on Foundations of Computer Science (FOCS)*, pages 575–584, 2010.

16 Shahar Chen, Dotan Di Castro, Zohar Karnin, Liane Lewin-Eytan, Joseph Seffi Naor, and Roy Schwartz. Correlated rounding of multiple uniform matroids and multi-label classification. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP)*, 2017.

17 Tasos C Christofides and Eutichia Vaggelatou. A connection between supermodular ordering and positive/negative association. *Journal of Multivariate analysis*, 88(1):138–151, 2004.

18 Jean-Claude Deville and Yves Tille. Unequal probability sampling without replacement through a splitting method. *Biometrika*, 85(1):89–101, 1998.

19 Devdatt Dubhashi, Johan Jonasson, and Desh Ranjan. Positive influence and negative dependence. *Combinatorics, Probability and Computing*, 16(01):29–41, 2007. doi:10.1017/S0963548306007772.

20 Devdatt Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *Random Struct. Algorithms*, 13(2):99–124, 1998. doi:10.1002/(SICI)1098-2418(199809)13:2\%3C99::AID-RSA1\%3E3.0.CO;2-M.

21 Devdatt P Dubhashi, Volker Priebe, and Desh Ranjan. Negative dependence through the FKG inequality. *BRICS Report Series*, 3(27), 1996.

22 Lawrence R Ernst. The maximization and minimization of sample overlap problems: a half century of results. In *Bulletin of the International Statistical Institute, Proceedings*, volume 58, pages 293–296, 1999.

23 Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. doi:10.1145/285055.285059.

24 Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991. doi:10.1016/0196-6774(91)90041-V.

25 Dongdong Ge, Simai He, Yinyu Ye, and Jiawei Zhang. Geometric rounding: a dependent randomized rounding scheme. *Journal of combinatorial optimization*, 22:699–725, 2011. doi:10.1007/S10878-010-9320-Z.

26 Thomas Holenstein. Parallel repetition: simplifications and the no-signaling case. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 411–419, 2007. doi:10.1145/1250790.1250852.

27 Kumar Joag-Dev and Frank Proschan. Negative association of random variables with applications. *The Annals of Statistics*, pages 286–295, 1983.

28 Nathan Keyfitz. Sampling with probabilities proportional to size: adjustment for changes in the probabilities. *Journal of the American Statistical Association*, 46(253):105–109, 1951.

29 Subhash Khot, Richard J Lipton, Evangelos Markakis, and Aranyak Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. In *Proceedings of the 1st Conference on Web and Internet Economics (WINE)*, pages 92–101, 2005. doi:10.1007/11600930_10.

30 Jon Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *Journal of the ACM (JACM)*, 49(5):616–639, 2002. doi:10.1145/585265.585268.

31 Josh Brown Kramer, Jonathan Cutler, and AJ Radcliffe. Negative dependence and srinivasan's sampling process. *Combinatorics, Probability and Computing*, 20(3):347–361, 2011. doi:10.1017/S0963548311000095.

32 Hongyang Liu and Yitong Yin. Simple parallel algorithms for single-site dynamics. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1431–1444, 2022. doi:10.1145/3519935.3519999.

33 Vahab Mirrokni, Michael Schapira, and Jan Vondrák. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In *Proceedings of the 9th ACM conference on Electronic commerce*, pages 70–77, 2008. doi:10.1145/1386790.1386805.

34 George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research (Math of OR)*, 3(3):177–188, 1978. doi:10.1287/MOOR.3.3.177.

35 Frederick Qiu and Sahil Singla. Submodular dominance and applications. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, 2022.

36 Ronald L Rivest. Symmetric encryption via keyrings and ecc. In *Northernmost Crypto Workshop, Longyearbyen, Norway, Midnight Lect*, 2016.

37 Ankit Sharma and Jan Vondrák. Multiway cut, pairwise realizable distributions, and descending thresholds. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 724–733, 2014. doi:10.1145/2591796.2591866.

38 Aravind Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *Proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS)*, pages 588–597, 2001. doi:10.1109/SFCS.2001.959935.