How to Send a Real Number Using a Single Bit (And Some Shared Randomness)

Ran Ben Basat
University College London, UK

Michael Mitzenmacher
Harvard University, Cambridge, MA, USA

Shay Vargaftik
VMware Research, Herzliya, Israel

Abstract

We consider the fundamental problem of communicating an estimate of a real number $x \in [0, 1]$ using a single bit. A sender that knows x chooses a value $X \in \{0, 1\}$ to transmit. In turn, a receiver estimates x based on the value of X. The goal is to minimize the cost, defined as the worst-case (over the choice of x) expected squared error.

We first overview common biased and unbiased estimation approaches and prove their optimality when no shared randomness is allowed. We then show how a small amount of shared randomness, which can be as low as a single bit, reduces the cost in both cases. Specifically, we derive lower bounds on the cost attainable by any algorithm with unrestricted use of shared randomness and propose optimal and near-optimal solutions that use a small number of shared random bits. Finally, we discuss open problems and future directions.

2012 ACM Subject Classification Theory of computation → Rounding techniques; Theory of computation → Stochastic approximation

Keywords and phrases Randomized Algorithms, Approximation Algorithms, Shared Randomness, Distributed Protocols, Estimation, Subtractive Dithering

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.25

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2010.02331`

Funding MM was supported in part by NSF grants CCF-1563710, CCF-1535795 and DMS-2023528. MM and RBB were supported in part by a gift to the Center for Research on Computation and Society at Harvard University.

Acknowledgements We thank the anonymous reviewers, Moshe Gabel, and Gal Mendelson for their helpful feedback and comments.

1 Introduction

We consider the fundamental problem of communicating an estimate of a real number $x \in [0, 1]$ using a single bit. A sender, that we call *Buffy*, knows x, and chooses a value $X \in \{0, 1\}$ to transmit. In turn, a receiver, that we call *Angel*, estimates x based on the value of X.

This problem naturally appears in distributed computations where multiple machines perform parallel tasks and transmit their results/state to an aggregator. If the bandwidth to the aggregator is limited, the machines must compress the data before sending it. Bandwidth optimization is fundamental in many domains, including network measurements [3, 11] and telemetry [5], load balancing [16, 22], and satellite communication [25]. We are especially motivated by recent work addressing the communication bottleneck in distributed and federated machine learning [13]. There, *clients* compute a local gradient and send it to a

parameter server that computes the global gradient and updates the model [15]. For the typical large-scale federated learning problems over edge devices (e.g., mobile phones), the devices may only be able to communicate a small number of bits per gradient coordinate. In fact, solutions such as 1-Bit SGD [20] and signSGD [6], have recently been studied as appealing low-communication solutions that use a single bit per coordinate. Another common communication-efficient solution is TernGrad [23] that quantizes each coordinate to $\{-1, 0, 1\}$ instead of $\{-1, 1\}$ as commonly done by the 1-bit algorithms.

It is often desirable that each estimate be an *unbiased* random variable with a mean equal to corresponding x. For example, this provides that the estimates' average is an unbiased estimate of the average value. Alternatively, there are cases where it is beneficial to allow *biased* estimates if it reduces the error for that setting (e.g., see EF-signSGD [14]).

In this work, we consider several variations of the above problem. For algorithms that provide unbiased estimates for every value x, we use the *worst-case* (over all values of x) variance as the cost function to be minimized. For biased estimates, we consider the worst-case *expected squared error* as the cost, as it coincides with variance for unbiased algorithms. That is, the worst-case is over the value of x, and the expectation of the cost is over the random choices used by the algorithm. Note that any lower bound for biased algorithms with these costs also applies to unbiased algorithms, and an upper bound for unbiased algorithms also applies to biased algorithms. We are interested in both lower and upper bounds in our work.

Beyond unbiased and biased variations, we also consider settings where Buffy and Angel have access to *shared randomness*. Shared randomness (often also referred to as public or common randomness) has been intensively studied in the field of communication complexity (e.g., see [17]). In our context, such shared randomness can arise naturally by having Buffy and Angel share a common seed for a pseudo-random number generator, for example. Here, we model the shared randomness as "perfectly random," leaving issues related to pseudo-randomness aside. Nevertheless, we consider solutions using limited amounts of shared randomness, including the case of just one bit of shared randomness. Such solutions may be easier and cheaper to implement, including with pseudo-random generators.

We remark that there are known approaches to this problem. These include (deterministic) rounding, randomized rounding (also called stochastic quantization), and subtractive dithering [18]. For a detailed survey of such techniques, we refer the reader to [9]. We discuss these methods and compare our results with them in context throughout the paper.

Our contribution. In this paper, we study how to minimize the cost (i.e., the worst-case variance or worst-case expected squared error) for various settings. First, we consider the setting where there is no shared randomness. In this setting, we show that randomized rounding is the optimal unbiased algorithm and that deterministic rounding is optimal when biased estimations are allowed. While these algorithms are widely used in practice, the optimality proofs under these cost models have not appeared elsewhere to the best of our knowledge.

Next, we explore how to reduce the cost if Buffy and Angel have access to shared randomness. We prove upper and lower bounds on the attainable variance for unbiased algorithms and expected squared error for biased ones. For our upper bounds, we assume that Buffy and Angel have access to ℓ shared random bits, for some $\ell \in \mathbb{N}$. We also consider the *limiting algorithms* where ℓ is not restricted. Our work addresses several extensions for cases where unbounded private randomness is allowed and when it is not. Finally, we consider the special case where x is known to be in $\{0, 1/2, 1\}$, a setting that is of high interest, for example, for the sign-based federated learning algorithms (e.g., [6, 14]) and particularly for TernGrad [23] that uses 3-level quantization. We provide an improved algorithm and a matching lower bound for this setting, thus proving its optimality. A summary of our results appears in Table 1.

Table 1 A summary of our results.

Scenario	Unbiased (Variance)	Biased (Exp. Squared Error)
No shared randomness	$1/4 = 0.25$ (randomized rounding) Optimal (Section 3.1)	$1/16 = 0.0625$ (deterministic rounding) Optimal (Section 3.2)
ℓ-bit shared randomness Unbounded private randomness	$\ell = 1 : \frac{1}{8} = 0.125$ $\ell = 8 : \frac{1}{12} + \frac{1}{393216} \approx 0.08334$ In general: $1/6 \cdot (1/2 + 4^{-\ell})$ (Section 5)	Open
ℓ-bit shared randomness No private randomness	Impossible (Section 6)	$\ell = 1 : \frac{1}{20} = 0.05$ $\ell = 8 : \approx 0.04599$ (Section 6.2.4)
Lower Bounds for $x \in [0, 1]$ Unbounded shared randomness	$1/16 = 0.0625$ (Section 4.2)	≈ 0.0459 (Section 4.1.2)
$x \in \{0, 1/2, 1\}$ 1-bit shared randomness No private randomness	$1/16 = 0.0625$ (Section 5)	$3/4 - 1/\sqrt{2} \approx 0.04289$ (Section 6.1)
Lower Bounds for $x \in \{0, 1/2, 1\}$ Unbounded shared randomness	$1/16 = 0.0625$ (Section 4.2)	$3/4 - 1/\sqrt{2} \approx 0.04289$ (Section 4)

2 Preliminaries

We start with some notation. We use $[n]$ to denote $\{0, 1, \ldots, n-1\}$, and $\Delta(S)$ to denote all possible probability distributions over the set S. (An element of $\Delta(S)$ will be expressed as a density function when S is uncountable, e.g., if $S = [0, 1]$.) We also use, for a binary predicate B, $\mathbb{1}_B$ as an indicator such that $\mathbb{1}_B = 1$ if B is true and 0 otherwise. Lastly, $\phi = (1 + \sqrt{5})/2$ denotes the Golden Ratio, which naturally comes up in some of our results.

Problem statement. Given a real number $x \in [0, 1]$, Buffy compresses it to a single bit value $X \in \{0, 1\}$ that is sent to Angel, who derives an estimate $\widehat{x}$. We also consider the special case where x is known to be in $\{0, 1/2, 1\}$. Our objective is to minimize the *cost* that is defined as the *worst-case expected squared error*, i.e., $\max_{x \in [0,1]} \mathbb{E}[(\widehat{x} - x)^2]$. Note that the worst-case is taken over the value of x and the expectation is over the randomness of the algorithm. In the *unbiased* setting, we additionally require $\mathbb{E}[\widehat{x}] = x$, in which case the cost becomes $\mathrm{Var}[\widehat{x}]$, i.e., the estimation variance. In some cases, we allow the parties to use ℓ bits of *shared randomness*. That is, we assume that they have access to a random value $h \in [2^\ell]$, known to both Buffy and Angel. When applicable, we use $r \in [0, 1]$ to denote the private randomness of Buffy.

3 Algorithms without Shared Randomness

We recap the performance of two standard algorithms – randomized and deterministic rounding. Interestingly, we show that when no shared randomness is allowed, randomized rounding is an optimal unbiased algorithm, and deterministic rounding is an optimal biased algorithm.

3.1 Randomized Rounding

In randomized rounding, Buffy uses private randomness to generate $X \sim \mathrm{Bernoulli}(x)$ which is sent using a single bit. In turn, Angel estimates $\widehat{x} = X$. Clearly, we have that $\mathbb{E}[\widehat{x}] = \mathbb{E}[X] = x$, and thus the algorithm is unbiased. The variance of the algorithm is $\mathrm{Var}[\widehat{x}] = \mathrm{Var}[X] = x(1 - x)$, and thus the worst-case is reached at $x = 1/2$, which gives a cost of $1/4$. The following theorem, whose proof is deferred to full version [4],

shows that randomized rounding is optimal, in the sense that no unbiased algorithm without shared randomness can have a worst-case variance lower than 1/4. Intuitively, requiring the estimate to be unbiased forces the algorithm to send 1 with a probability that is linear in x, maximizing its cost for $x = 1/2$. The proof also establishes the intuitive idea that it is not possible to benefit from randomness used solely by Angel.

▶ **Theorem 1.** *Any unbiased algorithm without shared randomness must have a worst-case variance of at least* 1/4.

3.2 Deterministic Rounding

With deterministic rounding, Buffy sends $X = 1$ when $x \geq 1/2$. Angel then estimates $\widehat{x} = X/2 + 1/4$. Deterministic rounding has an (absolute) error of at most 1/4, which is achieved for $x \in \{0, 1/2, 1\}$. Therefore, its cost is 1/16. The next theorem, whose proof appears in Supplement 9.1, shows that deterministic rounding is optimal, as no algorithm that does not use shared randomness can have a lower cost (even with unrestricted private randomness). We show that any such algorithm must have an expected squared error of at least 1/16 on at least one of $\{0, 1/2, 1\}$.

▶ **Theorem 2.** *Any algorithm without shared randomness must have a worst-case expected squared error of at least* 1/16.

4 Lower Bounds

We next explore lower bounds for algorithms with shared randomness. We use Yao's minimax principle [24] to prove a lower bound on the cost of any biased shared randomness protocol. Then, we show a stronger lower bound for unbiased algorithms using a different approach.

4.1 Lower Bound for Biased Algorithms

We place Yao's general formulation in the context of our specific problem.

▶ **Theorem 3** ([24])**.** *Consider our estimation problem over the inputs $x \in [0, 1]$, and let $\mathcal{A}$ be the set of all possible* deterministic *algorithms. For a (deterministic) algorithm $a \in \mathcal{A}$ and input $x \in [0, 1]$, let the function $c(a, x) = (a(x) - x)^2$ be its squared error.*
Then for any randomized *algorithm A and input distribution $q \in \Delta([0, 1])$ such that $X \sim q$:*

$$\max_{x\in[0,1]} \mathbb{E}\left[c(A, x)\right] \geq \min_{a\in\mathcal{A}} \mathbb{E}\left[c(a, X)\right] \ .$$

That is, the expected *squared error (over the choice of x from distribution q) of the* best *deterministic algorithm (for q) lower bounds the* expected *squared error of any randomized (potentially biased) algorithm A for the* worst-case *x (i.e., its cost). Further, the inequality holds as an equality for the optimal distribution q and algorithm A, i.e.,*

$$\min_{A} \max_{x\in[0,1]} \mathbb{E}\left[c(A, x)\right] = \max_{q} \min_{a\in\mathcal{A}} \mathbb{E}\left[c(a, X)\right] \ .$$

We proceed by selecting distributions q to lower bound $\min_{a\in\mathcal{A}} \mathbb{E}\left[c(a, X)\right]$. Notice that a deterministic algorithm can be defined using two values $v_0, v_1 \in [0, 1]$, such that if $|x-v_0| \leq |x-v_1|$ then Buffy sends 0 and Angel estimates x as v_0. Similarly, if $|x-v_0| > |x-v_1|$ then Buffy sends 1 and Angel estimates x as v_1.[1] In general, the above framework asserts that the cost, for the worst-case input, of any randomized algorithm is

[1] Other deterministic algorithms, e.g., that send 0 despite having $|x - v_0| > |x - v_1|$, can trivially be improved by an algorithm with the above form.

$$\max_{q\in\Delta([0,1])}\min_{v_0,v_1\in[0,1]}\int_0^1 \min\left\{(x-v_0)^2,(x-v_1)^2\right\}q(x)dx\ . \tag{1}$$

Our framework lower bounds the cost for any (biased or unbiased) algorithm that may use any amount of (shared or private) randomness. We now consider distributions q to lower bound the cost and later discuss the limitations of this approach.

4.1.1 The $\{0, 1/2, 1\}$ case

First, consider a discrete probability distribution q over $\{0, 1/2, 1\}$, and assume without loss of generality that $q(0) \le q(1)$. Any deterministic algorithm cannot estimate all values exactly, and it must map at least two of the points to a single value, thus allowing us to lower bound its cost. In Supplement 9.2, we prove the following.

▶ **Lemma 4.** *Any deterministic algorithm must incur a cost of at least* $\frac{q(0)\cdot q(1/2)}{4(q(0)+q(1/2))}$.

For $q(0) = q(1) = (2-\sqrt{2})/2$ and $q(1/2) = \sqrt{2}-1$, this lemma yields a lower bound of $3/4 - 1/\sqrt{2} \approx 0.04289$. In Section 6.1, we show that this is *an optimal* lower bound when x is known to be in $\{0, 1/2, 1\}$, by giving an algorithm with a matching cost.

4.1.2 The $[0, 1]$ case

In the general case, where x can take on any value in $[0, 1]$, we can get a tighter bound by looking at mixed distributions. Specifically, for parameters $a, w \in [0, 1/2]$, we consider the distribution where:

$$q(x) = \begin{cases} 0 & \text{with probability } w \\ 1 & \text{with probability } w \\ \text{uniform on } [a, 1-a] & \text{otherwise} \end{cases}\ .$$

Directly analyzing the optimal deterministic algorithm for this distribution proves complex. Instead, we first *hypothesize* that there exists an optimal deterministic algorithm for which either (1) $v_1 = 1 - v_0$ or (2) $v_1 = 1$. We emphasize that the lower bound holds even if the hypothesis is false. We then analyze what values of a, w maximize the cost of the best deterministic algorithm with the above form. Finally, we verify that the lower bound for the resulting distribution (with the specific a, w values) holds *for all deterministic algorithms.*

For case (1), we can express the cost as $2\cdot\left(wv_0^2 + \frac{1/2-w}{1/2-a}\int_a^{1/2}(x-v_0)^2dx\right)$. Similarly, for case (2), we get a cost of $wv_0^2 + \frac{1-2w}{1-2a}\cdot\left(\int_a^{\min\{1-a,(v_0+1)/2\}}(x-v_0)^2dx + \int_{(v_0+1)/2}^{1-a}(x-1)^2dx\right)$. Therefore, the cost of the optimal algorithm from the above family is given as:

$$\min\left\{2\cdot\left(wv_0^2 + \frac{1/2-w}{1/2-a}\int_a^{1/2}(x-v_0)^2dx\right),\right.$$
$$\left. wv_0^2 + \frac{1-2w}{1-2a}\cdot\left(\int_a^{\min\{1-a,(v_0+1)/2\}}(x-v_0)^2dx + \int_{(v_0+1)/2}^{1-a}(x-1)^2dx\right)\right\}\ .$$

This cost is maximized for $a = \frac{-2w^2+w-2\sqrt{w(1-w)}+1}{4w^2-6w+2}$, where the value of w satisfies

$$32w^3 - 56w^2 + \sqrt{w(1-w)}\cdot(8w^4 - 24w^3 + 38w^2 - 8w - 7) + 24w = 0.$$

The resulting bound is slightly larger than 0.0459. Next, we verify that for these a, w values, no deterministic algorithm can achieve a lower cost. Specifically, instead of using $q(x)$ as described above, we generate a finite discrete distribution. For a parameter $n \in \mathbb{N}$, we define:

$$q_n(x) = \begin{cases} \frac{\lfloor n \cdot w \rfloor}{n} & \text{if } x \in \{0,1\} \\ \frac{1}{n} & \text{if } x \in \left\{ a + \frac{1-2a}{2(n-2\lfloor n \cdot w \rfloor))} + i \cdot \frac{1-2a}{n-2\lfloor n \cdot w \rfloor} \mid i \in [n - 2\lfloor n \cdot w \rfloor] \right\} \\ 0 & \text{otherwise} \end{cases} .$$

Note that $\lim_{n\to\infty} q_n(x) = q(x)$. We then find the optimal deterministic solution for this distribution by using a deterministic k-means clustering algorithm (for $k = 2$), that is guaranteed to converge, e.g., using [10]. The code that we used to obtain this result is available at [2]. The optimal deterministic algorithm for $q_n(x)$ tends to have either $v_1 = 1 - v_0$ or $v_1 = 1$ as hypothesized. Finally, for $n = 10^6$ we get a cost higher than 0.0459 which we use as a lower bound.

We do not believe that this bound is tight. Nonetheless, as we show in Section 6.2.4, our bound is within 0.2% of the optimum.

4.2 Lower Bound for Unbiased Algorithms - Beyond MiniMax

We now consider lower bounds for unbiased algorithms. Utilizing Yao's lemma does not appear to provide means to obtain sharper bounds when requiring the algorithm to be unbiased. We consider the case where $x \in \{0, 1/2, 1\}$ and directly prove that any unbiased algorithm must have a worst-case variance of at least $1/16$. This lower bound then also holds for $x \in [0, 1]$, although an improved bound of $\pi^2/64 - 1/12 \approx 0.07$ for this case, based on an average-case analysis, is presented in [7].

Assume that we have $h \in [0, 1]$. Buffy sends $X(x, h)$ to Angel, which determines an estimate $\widehat{x}(X(x, h), h)$. For $x', x'' \in \{0, 1/2, 1\}$, let $p_{x',x''} = \Pr[X(x', h) = X(x'', h)]$ denote the probability (with respect to h) that the same bit is sent for x', x''. Since we send a single bit, we have that $p_{0,1/2} + p_{1/2,1} + p_{0,1} \geq 1$. For all $x', x'' \in \{0, 1/2, 1\}$, we define $H_{x',x''} = \{h \in [0, 1] : X(x', h) = X(x'', h)\}$ to be the set of shared-randomness values that would lead Buffy to send the same bit for both x' and x''. Next, denote by $G_{x',x''} = \mathbb{E}[\widehat{x} | h \in H_{x',x''}, x \in \{x', x''\}]$ the expected estimate value, conditioned on the shared randomness being in $H_{x',x''}$. We have that:

$$\begin{aligned} \mathrm{Var}[\widehat{x}|x = 0] &\geq p_{0,1/2} \cdot (G_{0,1/2} - 0)^2 + p_{0,1} \cdot (G_{0,1} - 0)^2 + p_{1/2,1} \cdot (\mathbb{E}[\widehat{x}|h \in H_{1/2,1}, x = 0] - 0)^2 \\ \mathrm{Var}[\widehat{x}|x = 1/2] &\geq p_{0,1/2} \cdot (G_{0,1/2} - 1/2)^2 + p_{0,1} \cdot (\mathbb{E}[\widehat{x}|h \in H_{0,1}, x = 1/2] - 1/2)^2 \\ &\qquad + p_{1/2,1} \cdot (G_{1/2,1} - 1/2)^2 \\ \mathrm{Var}[\widehat{x}|x = 1] &\geq p_{0,1/2} \cdot (\mathbb{E}[\widehat{x}|h \in H_{0,1/2}, x = 1] - 1)^2 + p_{0,1} \cdot (G_{0,1} - 1)^2 + p_{1/2,1} \cdot (G_{1/2,1} - 1)^2. \end{aligned}$$

To proceed, we require the algorithm to be unbiased:

$$\begin{aligned} &G_{0,1/2}p_{0,1/2} + G_{0,1}p_{0,1} + \mathbb{E}[\widehat{x}|h \in H_{1/2,1}, x = 0]p_{1/2,1} = 0 \\ &G_{0,1/2}p_{0,1/2} + \mathbb{E}[\widehat{x}|h \in H_{0,1}, x = 1/2]p_{0,1} + G_{1/2,1}p_{1/2,1} = 1/2 \\ &\mathbb{E}[\widehat{x}|h \in H_{0,1/2}, x = 1]p_{0,1/2} + G_{0,1}p_{0,1} + G_{1/2,1}p_{1/2,1} = 1. \end{aligned}$$

This allows us to express the expectations $\{E[\widehat{x}|h \in H_{x',x''}, x = x'''] | x', x'', x''' \in \{0, 1/2, 1\}\}$ using $p_{x',x''}, G_{x',x''}$ and obtain a set of three inequalities with six variables. Our full analysis, given in the full version [4], proceeds with a case analysis based on the value of $p_{0,1}$, the probability that the sender would send the same bit for $0, 1$. We show that there exists an optimal algorithm in which $p_{0,1/2} + p_{1/2,1} + p_{0,1} = 1$, $p_{0,1/2} = p_{1/2,1}$, and $G_{1/2,1} = 1 - G_{0,1/2}$. This reduces the number of variables to three, allowing us to optimize the expression and show a lower bound of $1/16$ on any unbiased algorithm.

5 Algorithms with Unbounded Private Randomness

Here, we consider the case where the shared randomness is limited to ℓ bits, i.e., $h \in [2^\ell]$, but Buffy may use unbounded private randomness $r \sim U[0,1]$ (that is independent of h).

We present the following algorithm: Buffy sends X to Angel, where

$$X \triangleq \begin{cases} 1 & \text{if } x \geq (r+h)2^{-\ell} \\ 0 & \text{otherwise} \end{cases}.$$

Angel then estimates $\widehat{x} = X + (h - 0.5(2^\ell - 1)) \cdot 2^{-\ell}$.

We first show that our protocol is unbiased. It holds that $\mathbb{E}[h] = 0.5(2^\ell - 1)$ and $(r+h) \sim U[0, 2^\ell]$ (i.e., $(r+h)2^{-\ell} \sim U[0,1]$), and thus $\mathbb{E}[\widehat{x}] = \mathbb{E}[X] = x$.
We state theorem, whose proof appears in the full version [4], that bounds the variance:

▶ **Theorem 5.** *Var*$[\widehat{x}] \leq 1/12 \cdot (1 - 4^{-\ell}) + 1/4 \cdot 4^{-\ell} = 1/6 \cdot (1/2 + 4^{-\ell})$.

In Supplement 9.3, we describe a simple generalization of this algorithm, together with a lower bound, for sending $k > 1$ bits. We now explain the connection to subtractive dithering and explore the applicability of the algorithm for the $x \in \{0, 1/2, 1\}$ special case.

Connection to subtractive dithering. First invented for improving the visibility of quantized pictures [18], subtractive dithering aims to alleviate potential distortions that originate from quantization. Subtractive dithering was later extended for other domains such as speech [8], distributed deep learning [1], and federated learning [21].

In our setting, subtractive dithering corresponds to using shared randomness to add *noise* ς to x before applying a deterministic quantization and subtracting ς from the estimation. Specifically, let $\mathcal{Q} : [0,1] \to \{0,1\}$ be a two-level deterministic quantizer such that $\mathcal{Q}(g) = 1$ if $g \geq 1/2$ and 0 otherwise. Then, in subtractive dithering Buffy sends $X = \mathcal{Q}(x + \varsigma)$ and Angel estimates $\widehat{x} = X - \varsigma$.

There are several noise classes that ς can be drawn from, as classified in [19], that yield $\widehat{x} \sim U[x - 1/2, x + 1/2]$. For example, ς can be distributed uniformly on $[-1/2, 1/2]$.

Consider our algorithm of this section without restricting the number of random bits (i.e., $\ell \to \infty$, and rescale so $h \in U[0,1]$). This would yield the following algorithm:

$$X \triangleq \begin{cases} 1 & \text{if } x \geq h \\ 0 & \text{otherwise} \end{cases}$$

and $\widehat{x} = X + h - 0.5$. Similarly to subtractive dithering, we get that $\widehat{x} \sim U[x - 1/2, x + 1/2]$, as we prove in the full version [4] for completeness. To see that the two algorithms are equivalent (for $\varsigma \sim U[-1/2, 1/2]$), denote $h' = 1/2 - h$ (i.e., $h' \sim U[-1/2, 1/2]$). Then $X = 1$ if $x + h' \geq 1/2$ and $\widehat{x} = X - h'$.

Therefore, we conclude that our algorithm provides a spectrum between randomized rounding ($\ell = 0$) and a form of subtractive dithering ($\ell \to \infty$). In practice, this means that a small number of shared random bits yields a variance that is close to that of subtractive dithering ($\text{Var}[\widehat{x}] = 1/12$). For example, with a single shared random byte (i.e., $\ell = 8$), our algorithm has a worst-case variance that is within 0.02% of 1/12.

The $x \in \{0, 1/2, 1\}$ case. Notice that if x is known to be in $\{0, 1/2, 1\}$, then our ($\ell = 1$) algorithm gives $\text{Var}[\widehat{x}] = 1/16$, as evident from Theorem 5. Further, in this case, we do not require the private randomness as we can rewrite Buffy's algorithm as:

$$X \triangleq \begin{cases} 0 & \text{if } x = 0 \\ 1-h & \text{if } x = 1/2 \\ 1 & \text{if } x = 1 \end{cases}$$

while Angel estimates $\widehat{x} = X + (h - 0.5)/2$. This algorithm considerably improves over randomized rounding (which is optimal when no shared randomness is allowed, as shown in the full version [4]), that has a variance of $1/4$ for $x = 1/2$; i.e., a single shared random bit reduces the worst-case variance by a factor of 4. Further, it also improves over subtractive dithering, reducing the variance by a $4/3$ factor. Finally, this result is optimal according to the Section 4.2 lower bound, even if unbounded shared randomness is allowed.

6 Algorithms without Private Randomness

In some cases, generating random bits may be expensive, e.g., when running on power-constrained devices. This is particularly acute when the device operates in an energy harvesting mode [26]. Past works have even considered how to "recycle" random bits (e.g., [12]). Therefore, it is important to study how to design algorithms that use just a few random bits. To address this need, we consider scenarios where Buffy and Angel have access to a shared ℓ-bit random value h, but no private randomness.

One thing to notice is that Angel can produce at most $2^{\ell+1}$ different values since Angel is deterministic after obtaining the $\ell + 1$ bits of h and X. In particular, this means there is no unbiased protocol for general $x \in [0, 1]$. Therefore, we focus on biased algorithms and study how shared randomness allows improving over deterministic rounding (which is optimal without shared randomness, as we show in Section 3.2).

We start by proposing an optimal algorithm for the case where x is known to be in $\{0, 1/2, 1\}$. Then, we present adaptations of the subtractive dithering estimation method for the biased $x \in [0, 1]$ setting. These improve over both (unbiased) subtractive dithering and deterministic rounding. To the best of our knowledge, these adaptations are novel. Next, we show how Buffy can further reduce the cost while, among other changes, using a small number of shared random bits. We conclude by giving design principles for numerically approximating the optimal algorithm and give realizations for small number of shared random bits.

6.1 The $x \in \{0, 1/2, 1\}$ Case

We now consider the scenario where x is guaranteed to be in $\{0, 1/2, 1\}$ using a single shared randomness bit $h \in \{0, 1\}$. For some $\alpha \in [0, 1]$, Buffy sends

$$X = \begin{cases} 1 & \text{if } x = 1 \vee (x = 1/2 \wedge h = 0) \\ 0 & \text{otherwise} \end{cases}$$

while Angel estimates $\widehat{x} = \alpha \cdot h + (1 - \alpha) \cdot X$.

For example, this means that if $x = 0$, the squared error is 0 if $h = 0$ and α^2 otherwise. That is, the expected squared error is $\alpha^2/2$. We optimize over the α value to minimize the cost

$$\min_{\alpha \in [0,1]} \max \left\{ \alpha^2/2, (1 - (1 - \alpha))^2/2, \mathbb{E}\left[(1/2 - (\alpha \cdot h + (1 - \alpha) \cdot (1 - h)))^2 \right] \right\}.$$

This is optimized for $\alpha = 1 - 1/\sqrt{2}$, yielding a cost of $3/4 - 1/\sqrt{2} \approx 0.04289$, which is *optimal* according to our Section 4 lower bound, even if unbounded shared randomness is allowed.

6.2 The $x \in [0,1]$ Case

An important observation regarding optimal biased algorithms is that they, without loss of generality, can be expressed as a pair of monotone increasing functions $T, Z_0 : [0,1] \to [0,1]$ as follows. Here T is a threshold function that determines whether 0 or 1 is sent, Z_0 is the estimator when 0 is received, and $Z_1 : [0,1] \to [0,1]$, given by $Z_1(h) = 1 - Z_0(1-h)$, is the estimator when 1 is received. That is, Buffy sends

$$X = \begin{cases} 1 & \text{if } x \geq T(h) \\ 0 & \text{otherwise} \end{cases}.$$

In turn, Angel estimates $\widehat{x} = Z_X(h)$. We further explain this representation in Supplement 9.4. Based on this observation, we next lay out a sequence of algorithmic improvements over deterministic rounding that leverage the shared randomness to reduce the cost. We visualize the algorithms resulting from each improvement in Figure 1.

6.2.1 Subtractive dithering adaptations

As subtractive dithering provides the lowest cost (albeit using unbounded shared randomness) of the previously mentioned unbiased algorithms, one may wonder if it is possible to adapt it to the biased scenario. Accordingly, we first briefly overview two natural adjustments that use unbounded shared randomness and improve over the $1/16$ cost of deterministic rounding. We then propose improved protocols that reduce the cost further despite using only a small number (e.g., $\ell = 3$) of random bits.

Intuitively, subtractive dithering may produce estimates that are outside the $[0,1]$ range. Therefore, by *truncating* the estimates to $[0,1]$ one may only reduce the expected squared error for any $x \neq 1/2$. However, it does not reduce the expected squared error for $x = 1/2$, and thus the cost would remain $1/12$.

To reduce the cost, one may further truncate the estimates to $[z, 1-z]$ for some $z \in [0, 1/2]$. Indeed, we show in the full version [4] that this truncation reduces the cost to ≈ 0.0602, for z satisfying $1/24 + z^2/2 + (2z^3)/3 = 0$ ($z \approx 0.17349$).

A better adaptation strategy is obtained by changing the estimation to a linear combination of X and h. Specifically, consider the protocol where Buffy sends (for a shared $h \sim U[0,1]$)

$$X = \begin{cases} 1 & \text{if } x \geq h \\ 0 & \text{otherwise} \end{cases}$$

and Angel estimates, for some $\alpha \in [0,1]$, $\alpha \cdot h + (1-\alpha) \cdot X$. Optimizing the parameters, we show in the full version [4] that this algorithm achieves a cost of $5/3 - \phi \approx 0.04863$, which is obtained for $\alpha = 2 - \phi \approx 0.382$. Interestingly, this cost is achieved for *all* $x \in [0,1]$.

In Figure 1, we illustrate this algorithm. As shown, the subtractive dithering adaption has $T(h) = h$ and $Z_0(h) = \alpha \cdot h$. This means that Buffy sends $X = 1$ if $x \geq h$ and Angel estimates $\widehat{x} = \alpha \cdot h$ if $X = 0$ and $\widehat{x} = (1 - \alpha \cdot (1-h)) = (1-\alpha) + \alpha \cdot h$ otherwise.

6.2.2 Deterministically rounding extreme values

We now show how to leverage a finite number of shared random bits ℓ to design improved algorithms. As we show, it is possible to benefit from *deterministically* rounding values that are "close" to 0 or 1 and use the shared randomness otherwise.

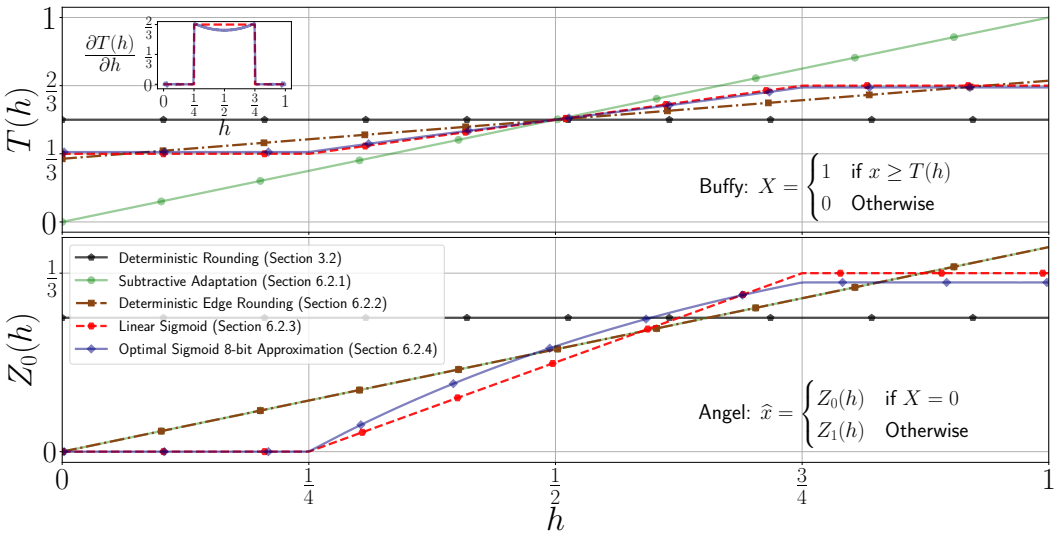

Figure 1 Illustration of the different biased algorithms. While deterministic rounding does not use the shared randomness and is thus constant, the other algorithms have both the threshold and estimation be monotone functions of h.

Similarly to the subtractive dithering adaptation above, Angel estimates x using a linear combination of h (with weight α) and X (with a weight of $1-\alpha$), where $\alpha \in [0,1]$ is chosen later. For all $i \in [2^\ell - 1]$, define the interval

$$\mathcal{I}_i = \left[(1-\alpha)/2 + i \cdot \frac{\alpha}{2^\ell - 1},\ (1-\alpha)/2 + (i+1) \cdot \frac{\alpha}{2^\ell - 1} \right). \tag{2}$$

In our algorithm, Buffy sends

$$X = \begin{cases} 0 & \text{if } x < (1-\alpha)/2 \\ \mathbb{1}_{h \leq i} & \text{if } x \in \mathcal{I}_i,\ i \in [2^\ell - 1] \\ 1 & \text{if } x \geq (1+\alpha)/2 \end{cases},$$

and Angel estimates: $\widehat{x} = \alpha \cdot h/(2^\ell - 1) + (1-\alpha) \cdot X$.

Note that we deterministically partition the range $[(1-\alpha)/2, (1+\alpha)/2]$ into $2^\ell - 1$ equally spaced intervals. Intuitively, the chosen intervals are designed to make the expected squared error a continuous function of x, as our analysis, given in the full version [4], indicates.

As we show, minimizing $cost = \min_\alpha \max_x \mathbb{E}[(\widehat{x} - x)^2]$ yields cumbersome expressions. For example, we get that with one shared random bit ($\ell = 1$), our algorithm has a cost of $1/18 \approx 0.05556$ (obtained for $\alpha = 1/3$, different than the α value used for the $x \in \{0, 1/2, 1\}$ case), lower than that of deterministic rounding (i.e., $1/16$). For $\ell = 2$, we obtain a cost of $\frac{259-140\sqrt{3}}{338} \approx 0.04885$ (reached for $\alpha = \frac{15-6\sqrt{3}}{13}$), and $\ell = 3$ bits further reduces the cost to $35/722 \approx 0.04848$ (when $\alpha = 7/19$). Additionally, with $\ell = 3$ bits, this improves over the subtractive dithering adaptions (that use unbounded shared randomness) for all $x \in [0,1]$. Notice that these costs are ≈21%, ≈6.4%, and ≈5.6% from the ≈ 0.0459 lower bound (see Section 4.1.2), and thus from the optimal algorithm. For completeness, we give the limiting algorithm (as $\ell \to \infty$) in the full version [4]. For intuition, we illustrate the limiting algorithm ($h \in [0,1]$) in Figure 1. As shown, we have $T(h) = \frac{1-\alpha}{2} + \alpha \cdot h$ (where $\alpha = 2 - \phi \approx 0.38$) and $Z_0(h) = \alpha \cdot h$. Observe that Angel uses the same estimation function as in Section 6.2.1,

but Buffy's threshold function is different. Intuitively, the new threshold function ensures that each x is mapped to the closest estimate value. For example, if $x = 0.1$ and $h = 0$, the subtractive adaptation would have $X = 1$ and thus $\widehat{x} = 1 - \alpha \approx 0.62$ while here we get $X = 0$ and $\widehat{x} = 0$.

Interestingly, the cost slightly and monotonically *increases* when increasing the number of bits ℓ beyond 3. This phenomenon suggests that we need more complex algorithms to leverage additional available random bits. We explore several approaches; in the full version [4], we show that by probabilistically selecting between the above algorithm (for $\ell \to \infty$) and the $\{0, 1/2, 1\}$ algorithm from Section 6.1, we can reduce the error to $\frac{6\sqrt{10}+11\sqrt{5}-18\sqrt{2}-17}{24} \approx 0.04644$. Intuitively, Buffy and Angel can implicitly agree on the chosen algorithm using the shared randomness. Here, we proceed by analyzing the potential benefits of non-uniform partitioning of the h values, which reduces the error further.

6.2.3 Non-uniform partitioning

Intuitively, the above algorithms have a threshold function that is linear in h; i.e., it takes the form $T(h) = a \cdot h + b$. We now show that this can be improved by looking at *sigmoid*-like functions. For ease of exposition, in this section, we consider $h \in [0, 1]$ to represent unbounded shared randomness, although the algorithm can be discretized given sufficient random bits. Recall from Section 6.2 that an algorithm can be expressed as a pair of functions $T, Z_0 : [0, 1] \to [0, 1]$ such that Buffy sends 1 if $x \geq T(h)$ while Angel estimates $Z_0(h)$ when receiving $X = 0$ and $Z_1(h) = 1 - Z_0(1 - h)$ otherwise. Here, we consider a *linear sigmoid* function (also illustrated in Figure 1), which, for some $h_0 \in [0, 1/2]$, is defined as

$$T(h) = \begin{cases} \alpha & \text{if } h < h_0 \\ \alpha + \frac{(1-2\alpha)(h-h_0)}{1-2h_0} & \text{if } h \in [h_0, 1-h_0] \\ 1 - \alpha & \text{otherwise} \end{cases}, \quad Z_0(h) = \begin{cases} 0 & \text{if } h < h_0 \\ \frac{(1-2\alpha)(h-h_0)}{1-2h_0} & \text{if } h \in [h_0, 1-h_0] \\ 1 - 2\alpha & \text{otherwise} \end{cases}.$$

Notice that in this algorithm we have $Z_0(h) = T(h) - \alpha$.

Our analysis, given in Supplement 9.5, shows that the cost is minimized for $h_0 = 1/4, \alpha = 1/3$, where the error is:

$$\mathbb{E}[(\widehat{x} - x)^2] = \mathbb{E}[(\widehat{x})^2] - 2x\mathbb{E}[\widehat{x}] + x^2 = \begin{cases} 5/108 - x/3 + x^2 & \text{if } x < 1/3 \\ 5/108 & \text{if } x \in [1/3, 2/3] \\ 77/108 - 5x/3 + x^2 & \text{otherwise} \end{cases}.$$

Therefore, the cost is $5/108 \approx 0.0463$, which is less than 0.9% higher than the 0.0459 lower bound (Section 4.1.2). The algorithm has two interesting properties. First, its expected squared error is constant for all $x \in \{0, 1\} \cup [1/3, 2/3]$ and, second, its expectation is not continuous as a function of x, as shown in Figure 2.

6.2.4 Towards the optimal algorithm

We now consider more general algorithms that have arbitrary estimate function Z_0. To that end, we use a numerical solver that approximates the optimal solution. Clearly, to define the input problem, we need to limit the number of variables and constraints. We achieve this using several observations:

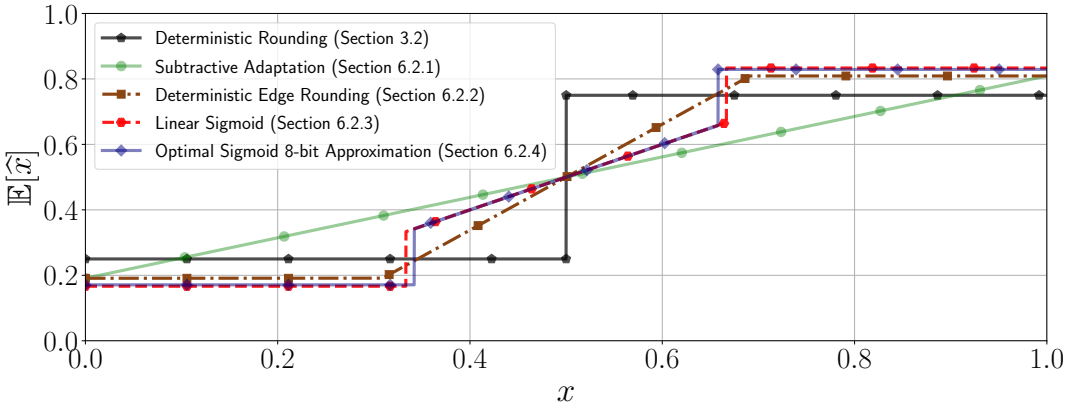

Figure 2 Illustration of the expectation of the different biased algorithms. Deterministic rounding does not use randomization and is therefore a step function, while others increase gradually in x. Notice that the expectations of the linear sigmoid and optimal approximation are not continuous.

- We consider bounded shared randomness $h \in [2^\ell]$ for $\ell \in \mathbb{N}$ bits. In fact, bounded shared randomness is precisely what allows us to develop this numerical approach.
- We use the observation that an optimal algorithm's T and Z_0 functions are not independent and satisfy $\forall h \in [0,1] : T(h) = \frac{Z_0(h)+Z_1(h)}{2} = \frac{Z_0(h)+(1-Z_0(1-h))}{2}$; this is because, that way, every $x \in [0,1]$ is estimated using the value closer to it between $Z_0(h)$ and $Z_1(h) = 1 - Z_0(1-h)$. In fact, the algorithms in sections 6.2.2-6.2.3 follow this rule, while the subtractive adaptation (Section 6.2.1) does not. As a result, we can define the variables $\{z_h | h \in [2^\ell]\}$ and derive the thresholds from the solver's output by $Z_0(h) = z_h$.
- For computing the maximal error for any $x \in [0,1]$, it is enough to look at a discrete set of points. This is because the number of possible estimates is $2^{\ell+1}$. Therefore, given two estimates $z_h, 1 - z_{2^\ell-1-h}$ that correspond to the values Angel uses given h and $X = 0$ or $X = 1$, the worst expected squared error (for this h) is obtained for $y_h \triangleq \frac{z_h+1-z_{2^\ell-1-h}}{2}$. Therefore, by checking all $x \in \{y_h \mid h \in [2^\ell]\}$, we can compute the cost.

Using these observations, we formulate the input as:

$$\begin{aligned} \underset{\{z_h | h\in[2^\ell]\}}{\text{minimize}} \quad & C \\ \text{subject to} \quad & C \geq \sum_{j=0}^{h} \left(y_h - (1 - z_{2^\ell-1-h})\right)^2 + \sum_{j=h+1}^{2^\ell-1} \left(y_h - z_h\right)^2, \quad h = 0, \ldots, 2^\ell - 1 \\ & y_h = \frac{z_h+1-z_{2^\ell-1-h}}{2}, \qquad z_h \in [0,1] \qquad\qquad h = 0, \ldots, 2^\ell - 1 \end{aligned}$$

In the above, we express the expected squared error at y_h by considering the h values for which $x \geq T(h)$ ($j \in [h]$) and those that $x < T(h)$. The output for the above problem does not seem to follow a compact representation. However, it is still possible to implement using a simple lookup table. For example, if $\ell = 8$, we can store all z_h when implementing Buffy and Angel. This algorithm's cost is lower than that of the linear sigmoid (that uses unbounded randomness) when using $\ell \geq 4$ bits. Specifically, using 4 shared random bits, the cost is ≈ 0.04611, while using 8 bits, it further reduces to ≈ 0.04599. Notice that these are less than 0.5% and 0.2% higher than the lower bound of Section 4.1.2. We note that this approach yields improvement even for a small number of shared random bits; for example, using $\ell = 1$ bit ($h \in \{0,1\}$), we get a cost of $1/20$ for $z_0 = 0.1, z_1 = 0.3$ which is equivalent to the following algorithm:

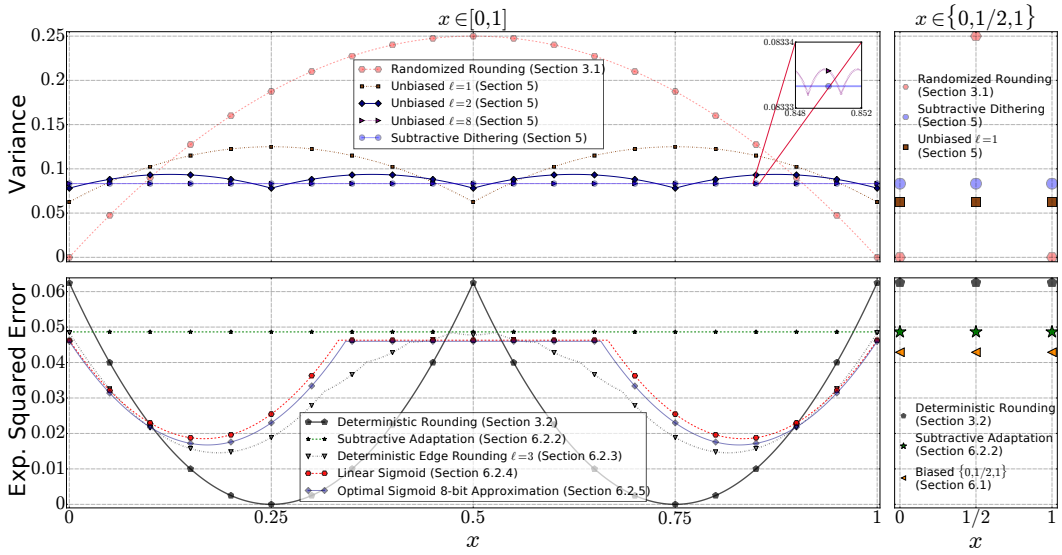

Figure 3 An illustration of the variance and expected squared error of the different algorithms. As shown, our unbiased algorithm is competitive with subtractive dithering despite using a single shared random byte, while our single-bit algorithm improves over subtractive on $\{0, 1/2, 1\}$. For the biased case, in addition to improving the $\{0, 1/2, 1\}$ case, our optimal sigmoid approximation algorithm achieves the lowest cost (less than 0.2% of the optimum!) while using a single shared random byte.

$$X = \begin{cases} 1 & \text{if } x \geq 0.4 + 0.2h \\ 0 & \text{otherwise} \end{cases}, \qquad \widehat{x} = 0.1 + 0.2h + 0.6X \ .$$

We visualize the resulting algorithm, for $\ell = 8$, in Figures 1 and 2. Notice that while the algorithm looks almost similar to our linear sigmoid, looking that the derivative $\frac{\partial T(h)}{\partial h}$ (Figure 1) shows that this optimal solution is not piece-wise linear.

7 Visual Comparison of the Algorithm Costs

We illustrate the various algorithms in Figure 3. In the unbiased case, notice how a single ($\ell = 1$) shared random bit significantly improves over randomized rounding (which is optimal when Buffy and Angel are restricted to private randomness). This further improves for larger ℓ values, where for $\ell = 8$ we have a cost that is only 0.02% higher than that of subtractive dithering, which uses unbounded shared randomness (the difference shown in zoom). When x is known to be in $\{0, 1/2, 1\}$ (right-hand side of the figure), it is evident how our unbiased $\ell = 1$ algorithm improves over both randomized rounding and subtractive dithering.

In the biased case, our adaptation to the subtractive dithering estimation (termed Subtractive Adaptation) improves over the cost of deterministic rounding. This is further improved by the algorithm of Section 6.2.2, termed Deterministic Edge Rounding, which is depicted using $\ell = 3$ bits as it minimizes its cost. Next, the Linear sigmoid (Section 6.2.3) shows how to lower the cost (using unbounded shared randomness) by non-uniform partitioning of the h values. Additionally, we show the optimal 8-bit algorithm (Section 6.2.4) that gets within 0.2% from the lower bound while using a single shared random byte. Finally, if x is known to be in $\{0, 1/2, 1\}$, our (optimal) biased $\{0, 1/2, 1\}$ algorithm improves over all other solutions while using only a single shared random bit.

8 Discussion

In this paper, we studied upper and lower bounds for the problem of sending a real number using a single bit. The goal is to minimize the cost, which is the worst-case variance for unbiased algorithms, or the worst-case expected squared error for biased ones. For all cases, we demonstrated how shared randomness helps to reduce the cost. Motivated by real-world applications, we derived algorithms with a bounded number of random bits that can be as low as a single shared bit. For example, in the unbiased case, using just one shared random bit reduces the variance two-fold compared to randomized rounding (which is optimal when no shared randomness is available). Further, using a single byte of shared randomness, our algorithm's variance is within 0.02% from the state of the art, which uses unbounded shared randomness. Our results are also near-optimal in the biased case, with a gap lower than 0.2% between the upper and lower bounds with a single shared random byte. Our upper bound is presented as a sequence of algorithms, each generalizing the previous while reducing the cost further.

For the special case where x is known to be in $\{0, 1/2, 1\}$, we give optimal unbiased and biased algorithms, together with matching lower bounds. Our algorithms use a single shared random bit, and the lower bounds show that the cost cannot be improved even when unbounded shared randomness is allowed.

We conclude by identifying directions for future research, beyond settling the correct bounds. First, our lower bounds apply for algorithms that use unbounded shared randomness, and new techniques for developing sharper bounds for other cases are of interest. Another direction is looking into optimizing the cost when sending k bits, for some $k > 1$. We make a first small step in Supplement 9.3, where we provide simple generalizations of our unbiased algorithm and lower bound to sending k bits. Finally, we are unclear on whether private randomness can help improve biased algorithms (see Table 1).

9 Supplementary Material

9.1 Optimality of Deterministic Rounding

We show that without shared randomness, deterministic rounding is an optimal biased solution. Notice that, in such a case, any protocol is defined by the probability of sending 1, denoted $Y(x)$, and the reconstruction distributions $V_0, V_1 \in \Delta([0,1])$.

Let us examine $\mathbb{E}[V_0]$ and $\mathbb{E}[V_1]$. We assume, without lost of generality, that $\mathbb{E}[V_0] \leq \mathbb{E}[V_1]$.

We have that:

$$\mathbb{E}[\widehat{x}] = Y(x)\mathbb{E}[V_1] + (1 - Y(x))\mathbb{E}[V_0].$$

That is, we have that *for any* $x \in [0,1]$: $\mathbb{E}[V_0] \leq \mathbb{E}[\widehat{x}] \leq \mathbb{E}[V_1]$. Next, we have that the cost, $\mathbb{E}[(\widehat{x} - x)^2]$, is bounded as

$$\mathbb{E}[(\widehat{x} - x)^2] \geq (\mathbb{E}[(\widehat{x} - x)])^2 .$$

In particular, for $x = 0$, we get that

$$\mathbb{E}[(\widehat{x} - x)^2 | x = 0] \geq (\mathbb{E}[\widehat{x} | x = 0])^2 \geq (\mathbb{E}[V_0])^2 .$$

Similarly, for $x = 1$, we have

$$\mathbb{E}[(\widehat{x} - x)^2 | x = 1] \geq (\mathbb{E}[(\widehat{x}) | x = 1] - 1)^2 \geq (1 - \mathbb{E}[V_1])^2 .$$

Notice that if $\mathbb{E}[V_0] \geq 0.25$ then $\mathbb{E}[(\widehat{x}-x)^2|x=0] \geq 1/16$, and similarly, if $\mathbb{E}[V_1] \leq 0.75$ then $\mathbb{E}[(\widehat{x}-x)^2|x=1] \geq 1/16$. Assume to the contrary that there exists an algorithm with a with a worst-case expected squared error lower than $1/16$, then we have $\mathbb{E}[V_0] \leq 0.25$ and $\mathbb{E}[V_1] \geq 0.75$. However, we have that $x = 0.5$ gives:

$$\begin{aligned}
\mathbb{E}[(\widehat{x}-x)^2|x=0.5] &= \mathbb{E}[\widehat{x}^2|x=0.5] - 2x\mathbb{E}[\widehat{x}|x=0.5] + 0.25 \\
&= Y(0.5)\mathbb{E}[V_1^2] + (1-Y(0.5))\mathbb{E}[V_0^2] - (Y(0.5)\mathbb{E}[V_1] + (1-Y(0.5))\mathbb{E}[V_0]) + 0.25 \\
&\geq Y(0.5)(\mathbb{E}[V_1])^2 + (1-Y(0.5))(\mathbb{E}[V_0])^2 - (Y(0.5)\mathbb{E}[V_1] + (1-Y(0.5))\mathbb{E}[V_0]) + 0.25 \\
&= Y(0.5)\cdot\mathbb{E}[V_1]\cdot(\mathbb{E}[V_1]-1) + (1-Y(0.5))\cdot\mathbb{E}[V_0]\cdot(\mathbb{E}[V_0]-1) + 0.25 \\
&\geq Y(0.5)\cdot 0.75\cdot(-0.25) + (1-Y(0.5))\cdot 0.25\cdot(-0.75) + 0.25 = 0.25 - 3/16 = 1/16.
\end{aligned}$$

In the first inequality, we used that fact that for any random variable V: $\mathbb{E}[V^2] \geq (\mathbb{E}[V])^2$, and in the second we used $\mathbb{E}[V_0] \leq 0.25$ and $\mathbb{E}[V_1] \geq 0.75$. This concludes the proof and establishes the optimality of deterministic rounding when no shared randomness is used.

9.2 Proof of the Biased $\{0, 1/2, 1\}$ Lower Bound

We recall Lemma 4:

▶ **Lemma 4.** *Any deterministic algorithm must incur a cost of at least* $\frac{q(0)\cdot q(1/2)}{4(q(0)+q(1/2))}$.

Proof. We denote by X_0 the set of values in $\{0, 1/2, 1\}$ that are closer to v_0 than to v_1. We assume without loss of generality that $v_0 \leq v_1$ and $q(0) \leq q(1)$ and prove that an optimal algorithm would set $v_0 = \frac{q(1/2)}{2(q(0)+q(1/2))}, v_1 = 1$, which incurs a cost of $\frac{q(0)\cdot q(1/2)}{4(q(0)+q(1/2))}$. Indeed, for this choice of v_0, v_1 we have that $X_0 = \{0, 1/2\}$, and we get a cost of

$$\begin{aligned}
q(0)\left(\frac{q(1/2)}{2(q(0)+q(1/2))}\right)^2 + q(1/2)\left(\frac{1}{2} - \frac{q(1/2)}{2(q(0)+q(1/2))}\right)^2 &= q(0)\left(\frac{q(1/2)}{2(q(0)+q(1/2))}\right)^2 \\
+ q(1/2)\left(\frac{q(0)}{2(q(0)+q(1/2))}\right)^2 &= \frac{q(0)q(1/2)^2 + q(1/2)q(0)^2}{4\left(q(0)+q(1/2)\right)^2} = \frac{q(0)\cdot q(1/2)}{4\left(q(0)+q(1/2)\right)}.
\end{aligned}$$

We now bound the performance of the optimal algorithm. We first notice that an optimal algorithm should have $0 \in X_0$ and $1 \notin X_0$. Next, notice that v_0 should be at most $1/2$ and v_1 should be at least $1/2$. Otherwise, one can improve the error for $x = 0$ or $x = 1$, respectively, without increasing the error at $1/2$. Further, observe that an optimal algorithm must have $v_0 = 0$ or $v_1 = 1$. That is because if $1/2 \in X_0$, we can reduce the error for $x = 1$ by setting $v_1 = 1$. Similarly, when $1/2 \notin X_0$, choosing $v_0 = 0$ decreases the error for $x = 0$. Now, we claim that there exists an optimal algorithm for which $v_1 = 1$. Consider some solution, and set $v_0' = 1 - v_1$ and $v_1' = 1$. This does not affect the error of $x = 1/2$, and does not increase the cost as $q(0) \leq q(1)$. We are left with choosing v_0; let us denote by $c(v_0) = q(0)v_0^2 + q(1/2)(1/2 - v_0)^2$ the resulting cost. This function has a minimum at $v_0 = \frac{q(1/2)}{2(q(0)+q(1/2))}$, which gives a cost of $\frac{q(0)\cdot q(1/2)}{4(q(0)+q(1/2))}$. ◀

This cost is maximized for $q(1/2) = \sqrt{2} - 1$ and $q(0) = q(1) = \frac{2-\sqrt{2}}{2}$, giving a lower bound of $3/4 - 1/\sqrt{2} \approx 0.04289$. In fact, one can verify that this is the best attainable lower bound for *any* discrete distribution on three points. Further, in Section 6.1, we show that this is *an optimal* lower bound when x is known to be in $\{0, 1/2, 1\}$, by giving an algorithm with a matching cost.

9.3 Generalization to k Bits

9.3.1 General Quantized Algorithm

We use a hash function h such that $h \in \{0,1\}^\ell$ is uniformly distributed. Let $A \sim U[0,1]$ be independent of h.

$$C = \lfloor (2^k - 1) \cdot x \rfloor$$

$$p = (2^k - 1) \cdot x - \lfloor (2^k - 1) \cdot x \rfloor$$

$$R = 2^k - 1$$

We then set

$$X \triangleq \begin{cases} C+1 & \text{if } p \geq (A+h)2^{-\ell} \\ C & \text{otherwise} \end{cases}$$

We send X to Angel which estimates

$$\widehat{x} = \frac{X + (h - 0.5(2^\ell - 1)) \cdot 2^{-\ell}}{R}.$$

To show that our protocol is unbiased, notice that: $\mathbb{E}[X] = R \cdot x$ and that $\mathbb{E}[h] = 0.5(2^\ell - 1)$.

9.3.2 Lower Bounds

Similarly to the 1-bit case, we consider the discrete distribution over

$$\left\{ i \cdot \left(\frac{1}{3 \cdot 2^{k-1} - 1} \right) \mid i \in \left\{0, 1, \ldots, 3 \cdot 2^{k-1} - 1\right\} \right\}.$$

We set $a_{1/2} = \frac{\sqrt{2}-1}{2^{k-1}}$ and $a_0 = a_1 = \frac{1 - a_{1/2}}{2^k}$ and

$$\forall i : q\left(i \cdot \left(\frac{1}{3 \cdot 2^{k-1} - 1} \right) \right) = a_{(i \mod 3)/2}.$$

When each consecutive set of three points has the same probability, one can derive an optimal algorithm with precisely two values between each such triplet. The optimal choice of locations of the values in each triplet is similar to our single-bit analysis of the previous subsection, i.e., one should have a values at

$$\left\{ \frac{\sqrt{2} - 1 + i}{2^{k-1}} \;\middle|\; i \in \left\{0, 1, \ldots, 2^{k-1} - 1\right\} \right\} \bigcup \left\{ \frac{i}{2^{k-1}} \middle|\; i \in \left\{1, \ldots, 2^{k-1}\right\} \right\}.$$

We turn into calculating the cost. Notice that every triplet has a width of $\frac{2}{3 \cdot 2^{k-1} - 1}$. Therefore, the cost now reduces, compared to the 1-bit analysis, by a factor of $\left(\frac{2}{3 \cdot 2^{k-1} - 1} \right)^2$. That is, we get a lower bound of $\frac{3 - 2\sqrt{2}}{(3 \cdot 2^{k-1} - 1)^2} = \frac{3 - 2\sqrt{2}}{2.25(2^k - 2/3)^2}$. We note that, for large k and ℓ values, our variance is within 10% of the lower bound, as

$$\lim_{k,\ell \to \infty} \frac{\frac{1}{12(2^k - 1)^2}}{\frac{3 - 2\sqrt{2}}{2.25(2^k - 2/3)^2}} = \frac{9 + 6\sqrt{2}}{16} \approx 1.093.$$

9.4 Reducing Algorithms to Monotone T, Z_0 Functions

We now show how an algorithm can be represented as described in Section 6.2. Fixing the shared randomness value h, Angel estimates $\widehat{x}$ solely based on the sent bit X; denote these values by $A_X(h)$. Without loss of generality, assume that $\forall h : A_1(h) \geq A_0(h)$.[2] This means that, in an optimal algorithm, Buffy should send $X = 1$ if $x \geq \frac{A_0(h)+A_1(h)}{2}$, as otherwise the error would be suboptimal for any x not satisfying the condition. In particular, this means that we can express Buffy's algorithm using a threshold function T.

Next, we claim that the threshold function can be considered monotone, without increasing the cost. To that end, we first consider a finite shared randomness $h \in [2^\ell]$. In such a case, if there exists some $h_1 > h_2 \in [2^\ell]$ such that $T(h_1) < T(h_2)$, we can modify the algorithm as follows: For all $h \notin \{h_1, h_2\}$, no modification is made. If $h = h_1$, then the modified algorithm works as if $h = h_2$, and vice versa. Following this process, we can sort T until it becomes monotone. A similar argument can be made for the continuous ($h \in [0, 1]$) case (possibly with an additional ϵ discretization cost).

We proceed with showing that there exists an optimal algorithm in which $Z_1(h) = 1 - Z_0(1 - h)$. This is achieved using a symmetry observation. Specifically, if an algorithm does not satisfy the above, consider its "dual algorithm": instead of sending x using $T(h)$, we send $x' = 1 - x$ using $T'(h) = 1 - T(1 - h)$; similarly, Angel estimates $\widehat{x}' = 1 - \widehat{x}$. Then, if both Buffy and Angel use the shared randomness to implicitly agree on whether to run the original or dual algorithms, each with probability half, the cost can only decrease. Additional details are given in the full version [4].

9.5 Analysis of the Linear Sigmoid (Section 6.2.3)

First, we have (see Section 6.2) that the estimate function for $X = 1$ is:

$$Z_1(h) = 1 - Z_0(1-h) = \begin{cases} 2\alpha & \text{if } h < h_0 \\ 2\alpha + (1-2\alpha) \cdot \frac{h-h_0}{1-2h_0} & \text{if } h \in [h_0, 1-h_0] \\ 1 & \text{otherwise} \end{cases}.$$

For $x \in [\alpha, 1-\alpha]$, denote by $T^{-1}(x) = \frac{(1-2h_0)(x-\alpha)}{1-2\alpha} + h_0$ the value such that $T(T^{-1}(x)) = x$. We proceed with computing the expectation:

$$\mathbb{E}[\widehat{x}|x < \alpha](\implies X = 0) = h_0 \cdot (1-2\alpha) + \int_{h_0}^{1-h_0} \left((1-2\alpha) \cdot \frac{h-h_0}{1-2h_0} \right) dh = 1/2 - \alpha$$

$$\begin{aligned} \mathbb{E}[\widehat{x}|x > 1-\alpha](\implies X = 1) &= h_0 \cdot (1+2\alpha) + \int_{h_0}^{1-h_0} \left(2\alpha + (1-2\alpha) \cdot \frac{h-h_0}{1-2h_0} \right) dh \\ &= 1/2 + \alpha \end{aligned}$$

[2] If for some h, $A_0(h) > A_1(h)$, there exists an equivalent algorithm that replaces the role of $X = 0$ and $X = 1$ for this specific h.

$$\begin{aligned}\mathbb{E}[\widehat{x}|x\in[\alpha,1-\alpha]] &= h_0\cdot(2\alpha)+h_0(1-2\alpha)\\ &\quad+\int_{h_0}^{T^{-1}(x)}\left(2\alpha+(1-2\alpha)\cdot\frac{h-h_0}{1-2h_0}\right)dh\\ &\quad+\int_{T^{-1}(x)}^{1-h_0}\left((1-2\alpha)\cdot\frac{h-h_0}{1-2h_0}\right)dh\\ &= h_0+2\alpha(T^{-1}(x)-h_0)+\int_{h_0}^{1-h_0}\left((1-2\alpha)\cdot\frac{h-h_0}{1-2h_0}\right)dh\\ &= 1/2+(-1+2h_0)\alpha+2\alpha(T^{-1}(x)-h_0)=1/2-\alpha+2\alpha T^{-1}(x)\end{aligned}$$

Therefore, we have:

$$\mathbb{E}[\widehat{x}]=\begin{cases}1/2-\alpha & \text{if } x<\alpha\\ 1/2-\alpha+2\alpha\left(\frac{(1-2h_0)(x-\alpha)}{1-2\alpha}+h_0\right) & \text{if } x\in[\alpha,1-\alpha]\\ 1/2+\alpha & \text{otherwise}\end{cases}.$$

Next, we calculate the second moment of the estimate:

$$\begin{aligned}\mathbb{E}[(\widehat{x})^2|x<\alpha](\implies X=0) &= h_0\cdot(1-2\alpha)^2+\int_{h_0}^{1-h_0}\left((1-2\alpha)\cdot\frac{h-h_0}{1-2h_0}\right)^2dh\\ &= 1/3+h_0/3-4\alpha/3-4h_0\alpha/3+4\alpha^2/3+4h_0\alpha^2/3\end{aligned}$$

$$\begin{aligned}\mathbb{E}[(\widehat{x})^2|x>1-\alpha](\implies X=1) &= h_0\cdot(1+(2\alpha)^2)\\ &\quad+\int_{h_0}^{1-h_0}\left(2\alpha+(1-2\alpha)\cdot\frac{h-h_0}{1-2h_0}\right)^2dh\\ &= 1/3+h_0/3+2\alpha/3-4h_0\alpha/3+4\alpha^2/3+4h_0\alpha^2/3\end{aligned}$$

$$\begin{aligned}\mathbb{E}[(\widehat{x})^2|x\in[\alpha,1-\alpha]] &= h_0\cdot((1-2\alpha)^2+(2\alpha)^2)\\ &\quad+\int_{h_0}^{T^{-1}(x)}\left(2\alpha+(1-2\alpha)\cdot\frac{h-h_0}{1-2h_0}\right)^2dh+\int_{T^{-1}(x)}^{1-h_0}\left((1-2\alpha)\cdot\frac{h-h_0}{1-2h_0}\right)^2dh\\ &= h_0\cdot((1-2\alpha)^2+(2\alpha)^2)+\frac{(1-2h_0)(x-\alpha)(x^2+4x\alpha+7\alpha^2)}{3-6\alpha}\\ &\quad+\frac{(2h_0-1)(-1+x+\alpha)(1+x+x^2-4x\alpha+\alpha(-5+7\alpha))}{3-6\alpha}\\ &= \frac{1-6\alpha-6\alpha x^2(-1+2h_0)+12\alpha^2-14\alpha^3+h_0(1-6\alpha+24\alpha^2-20\alpha^3)}{3-6\alpha}\end{aligned}$$

Putting it together, we get:

$$\mathbb{E}[\widehat{x}^2]=\begin{cases}1/3+h_0/3-4\alpha/3-4h_0\alpha/3+4\alpha^2/3+4h_0\alpha^2/3 & \text{if } x<\alpha\\ \frac{1-6\alpha-6\alpha x^2(-1+2h_0)+12\alpha^2-14\alpha^3+h_0(1-6\alpha+24\alpha^2-20\alpha^3)}{3-6\alpha} & \text{if } x\in[\alpha,1-\alpha]\\ 1/3+h_0/3+2\alpha/3-4h_0\alpha/3+4\alpha^2/3+4h_0\alpha^2/3 & \text{otherwise}\end{cases}.$$

By solving

$$\min_{h_0\in[0,1/2],\alpha\in[0,1/2]}\max_{x\in[0,1]}\mathbb{E}[(\widehat{x}-x)^2]=\min_{h_0\in[0,1/2],\alpha\in[0,1/2]}\max_{x\in[0,1]}\mathbb{E}[\widehat{x}^2]-2x\mathbb{E}[\widehat{x}]+x^2,$$

we get that the algorithm is optimized for $\alpha=1/3$ and $h_0=1/4$, where the resulting cost is:

$$\mathbb{E}[(\widehat{x}-x)^2] = \mathbb{E}[(\widehat{x})^2] - 2x\mathbb{E}[\widehat{x}] + x^2$$
$$= \begin{cases} 5/108 - x/3 + x^2 & \text{if } x < 1/3 \\ 5/108 & \text{if } x \in [1/3, 2/3] \\ 77/108 - 5x/3 + x^2 & \text{otherwise} \end{cases}.$$

Therefore, the cost is $5/108 \approx 0.0463$, which is less than 0.9% higher than the 0.0459 lower bound (Section 4.1.2).

References

1. Afshin Abdi and Faramarz Fekri. Indirect Stochastic Gradient Quantization and Its Application in Distributed Deep Learning. In *AAAI*, 2020.
2. Ran Ben Basat, Michael Mitzenmacher, and Shay Vargaftik. Biased [0,1] proof. URL: `https://gist.github.com/ranbenbasat/9959d1c70471fe870424eefbecd3e13c`.
3. Ran Ben-Basat, Gil Einziger, Isaac Keslassy, Ariel Orda, Shay Vargaftik, and Erez Waisbard. Memento: Making Sliding Windows Efficient for Heavy Hitters. In *ACM CoNEXT*, 2018.
4. Ran Ben-Basat, Michael Mitzenmacher, and Shay Vargaftik. How to Send a Real Number Using a Single Bit (and Some Shared Randomness). *CoRR*, abs/2010.02331, 2020. `arXiv:2010.02331`.
5. Ran Ben Basat, Sivaramakrishnan Ramanathan, Yuliang Li, Gianni Antichi, Minian Yu, and Michael Mitzenmacher. PINT: Probabilistic In-band Network Telemetry. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 662–680, 2020.
6. Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed Optimisation for Non-Convex Problems. In *International Conference on Machine Learning*, pages 560–569, 2018.
7. Zarathustra Elessar Brady (`https://cstheory.stackexchange.com/users/50608/zeb`). Is Subtractive Dithering the Optimal Algorithm for Sending a Real Number Using One Bit? URL: `https://cstheory.stackexchange.com/questions/48281`.
8. MR Garey, David Johnson, and Hans Witsenhausen. The complexity of the generalized Lloyd-Max problem (Corresp.). *IEEE Transactions on Information Theory*, 28(2):255–256, 1982.
9. Robert M. Gray and David L. Neuhoff. Quantization. *IEEE transactions on information theory*, 44(6):2325–2383, 1998.
10. Allan Grønlund, Kasper Green Larsen, Alexander Mathiasen, Jesper Sindahl Nielsen, Stefan Schneider, and Mingzhou Song. Fast Exact K-Means, K-Medians and Bregman Divergence Clustering in 1D. *arXiv preprint*, 2017. `arXiv:1701.07204`.
11. Rob Harrison, Qizhe Cai, Arpit Gupta, and Jennifer Rexford. Network-Wide Heavy Hitter Detection with Commodity Switches. In *Proceedings of the Symposium on SDN Research*, pages 1–7, 2018.
12. Russell Impagliazzo and David Zuckerman. How to Recycle Random Bits. In *FOCS*, volume 30, pages 248–253, 1989.
13. Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and Open Problems in Federated Learning, 2019. `arXiv:1912.04977`.

14 Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error Feedback Fixes SignSGD and other Gradient Compression Schemes. In *International Conference on Machine Learning*, pages 3252–3261, 2019.

15 Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated Optimization: Distributed Optimization Beyond the Datacenter. *arXiv preprint*, 2015. `arXiv:1511.03575`.

16 Michael Mitzenmacher. Queues with Small Advice. *arXiv preprint*, 2020. `arXiv:2006.15463`.

17 Ilan Newman. Private vs. Common Random bits in Communication Complexity. *Information processing letters*, 39(2):67–71, 1991.

18 Lawrence Roberts. Picture Coding Using Pseudo-Random Noise. *IRE Transactions on Information Theory*, 8(2):145–154, 1962.

19 Leonard Schuchman. Dither Signals and Their Effect on Quantization Noise. *IEEE Transactions on Communication Technology*, 12(4):162–165, 1964.

20 Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-Bit Stochastic Gradient Descent and its Application to Data-Parallel Distributed Training of Speech DNNs. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

21 Nir Shlezinger, Mingzhe Chen, Yonina C Eldar, H Vincent Poor, and Shuguang Cui. UVeQFed: Universal Vector Quantization for Federated Learning. *arXiv preprint*, 2020. `arXiv:2006.03262`.

22 Shay Vargaftik, Isaac Keslassy, and Ariel Orda. LSQ: Load Balancing In Large-Scale Heterogeneous Systems With Multiple Dispatchers. *IEEE/ACM Transactions on Networking*, 28(3):1186–1198, 2020.

23 Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning. In *Advances in neural information processing systems*, pages 1509–1519, 2017.

24 Andrew Chi-Chin Yao. Probabilistic Computations: Toward a Unified Measure of Complexity. In *IEEE FOCS*, 1977.

25 Guoxia Yu, Tanya Vladimirova, and Martin N Sweeting. Image Compression Systems on Board Satellites. *Acta Astronautica*, 64(9-10):988–1005, 2009.

26 Pengyu Zhang and Deepak Ganesan. Enabling Bit-by-Bit Backscatter Communication in Severe Energy Harvesting Environments. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pages 345–357, 2014.