Update Efficiency and Local Repairability
Limits for Capacity Approaching Codes

Arya Mazumdar Member, IEEE, Venkat Chandar, and Gregory W. Wornell Fellow, IEEE

Abstract

An update-efficient code is a mapping from messages to codewords such that for a small perturbation in the message the corresponding codeword changes only slightly. Analogously, a code is called locally recoverable or repairable if any symbol of a codeword can be recovered by reading only a small (constant) number of other symbols. The notions of local recoverability and update-efficiency are important in the area of distributed storage systems, where the most frequent error event is a single storage node failure and most updates on data are small. A common objective is to repair a failed node by downloading data from as few other storage nodes as possible. For updates, one wants to change as few nodes as possible.

In this paper, we first study update-efficient error-correcting codes and their basic properties. While update-efficiency and error-correction are two conflicting properties, we provide conditions for the existence of such codes. One of our main results is to show that the update-efficiency has to scale logarithmically with the block-length of the code if we are to achieve any nontrivial rate with vanishing...
probability of error over the binary symmetric or binary erasure channels. There exist capacity-achieving codes with this scaling. We also explore the notion of update-efficiency in the presence of adversarial errors.

The capacity result for the case of locally repairable codes is also considered here. We provide tight upper and lower bounds on the local-recoverability of a code that achieves capacity on the binary erasure channel. In particular, it is shown that if the code-rate is \( \epsilon \) less than the capacity, then for the optimal codes, the maximum number of codeword symbols required to recover one lost symbol must scale as \( \log 1/\epsilon \).

Although most of the results in this paper are presented assuming a binary alphabet, with little effort, the results extend to the case of larger alphabets, the usual scenario in distributed storage.

I. Introduction

A. Update efficiency

In a distributed storage system, like most other scenarios in communication, the data stored is susceptible to errors. Most frequently these errors are in the form of storage-node (server) failures. Therefore, before writing the data in memory, usually it is encoded with an error-correcting code or channel code. Erasure channels are a common failure model considered in these systems.

At the same time, data to be stored in a storage network is frequently changing. Updating the coded data-packets stored in servers consumes bandwidth and energy. If the change in the data is significant, then updating a linear (in blocklength of code) number of bits is unavoidable. However, it is of interest to construct codes that have sub-linear update time for small changes in message (data).

To capture this scenario, the notion of update-efficiency for channel codes was introduced in [2]. A code is called update-efficient if, for any small change in the message, the corresponding codeword changes only slightly. In particular, [2] considers codes over binary alphabet and shows the existence of a code that achieves the capacity of the binary erasure channel (BEC) [8 §7.1.5] with the following property. For any 1 bit change in the message, a logarithmic (in blocklength) number of bits need to be updated in the codeword.
In a follow-up paper [26], the application of update-efficient codes to distributed storage is explored further. Using the randomized codes proposed in [2], this paper shows that it is possible to have both update-efficiency and repair-bandwidth efficiency, a property desirable in distributed storage, with a code that achieves the optimal rate for a binary erasure channel. In another recent paper [17], update-efficiency of random linear codes is studied.

In this paper, we first show that one of the main propositions of [2] can be very easily derived. That is, there exist linear codes that achieve the capacity of the binary erasure channel and for any 1 bit change in the message, only logarithmic number of bits have to be updated in the codeword. Our main result, however, is the converse statement. Namely, we show that for some $\alpha > 0$, there cannot exist a linear code that has both positive rate and arbitrarily small probability of error with the following property: for any single bit change in the message, fewer than $\alpha \log n$ bits need to be updated in the codeword, where $n$ is the blocklength.

In addition to erasures, we also study update-efficient codes that correct errors. This is applicable to distributed storage when the failure model involves errors instead of erasures. An additional potential application of these codes is the storage or display of video over a noisy communication link. As the messages (video-frames) change only a little from one frame to the next, an update-efficient code can greatly reduce the overhead of energy and time.

Correcting errors is generally a more difficult task than correcting erasures. We show that our aforementioned results regarding BEC also extends to the binary symmetric channel (BSC) [8, §7.1.4]. Namely, there exist linear codes that have a rate arbitrarily close to the capacity of BSC and an arbitrarily small probability of error, with the property that for any one bit change in message a logarithmic number of bits need to be updated in the codeword. Again the more interesting part is the converse result that holds true for all codes—linear or nonlinear. We further determine $\alpha > 0$ such that there cannot exist a linear code with positive rate and arbitrarily small probability of error that requires fewer than $\alpha \log n$ bit updates in the codeword if a single bit changes in the message, where $n$ is the blocklength.

We also briefly focus on update-efficient codes that can correct some arbitrary set of errors or adversarial errors/erasures. The model of adversarial erasures in the channel is common in current
storage applications. For the case of adversarial errors, update-efficiency and error correction are two directly conflicting properties. Namely, it is impossible to correct more than \( \sim t/2 \) errors (or \( \sim t \) erasures) with a code that needs at most \( t \) bits of update for any one bit change in the message. This is true because the minimum pairwise distance between the codewords (i.e., \textit{minimum distance}) is upper bounded by \( t \). We discuss several properties of linear codes that are useful for constructing good update-efficient adversarial error correcting codes, i.e., codes where this bound is achieved. Most important observation in this case is, perhaps, that if there exists a linear code, then there will exist another code with same parameters where the bound is achieved. We also extend the definition of update-efficiency from [2].

Following [2] and [21], it can be shown that there exist codes with rate equal to the capacity of BSC with crossover probability \( p \), that correct any \( \sim pn \) adversarial errors, \( n \) blocklength, and need at most logarithmic (in \( n \)) bits of update for any one bit change in the message, provided that there is sufficient shared randomness between the encoder and the decoder.

B. Local recovery

The notion of \textit{local recovery} for a code is, in some sense, dual to the notion of update-efficiency of a code. In a locally recoverable code \( \mathcal{C} \), any codeword symbol of \( \mathcal{C} \) can be recovered from at most a constant number of other symbols of the codeword. This property is desirable in distributed storage systems, and was introduced in that context in [11]. In [11], as well as in [25], locally recoverable codes that also correct a number of adversarial errors (or erasures) were considered. A trade-off between the local recoverability and error-correction was presented. In particular, it was shown that for a \( q \)-ary linear code, \( q \geq 2 \), of blocklength \( n \),

\[
d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2,
\]

where \( d \) is the minimum distance, \( k \) is the dimension, and \( r \) is the local recoverability (maximum number of symbols required to reconstruct one symbol) of the code. This can be generalized to nonlinear codes with all possible alphabet sizes. Indeed, it is shown in [5] that for any \( q \)-ary
code with size \( M \), local recoverability \( r \), and minimum distance \( d \),

\[
\log M \leq \min_{1 \leq t \leq \left\lceil \frac{n}{r+1} \right\rceil} \left[ t r + \log A_q(n - t(r + 1), d) \right],
\]

(1)

where \( A_q(n, d) \) is the maximum size of a \( q \)-ary code of length \( n \) and distance \( d \). It should be noted that the local recovery property does not have immediate conflict with the error-correction property (as opposed to update-efficiency, which requires a number of codewords to be close to each other).

Locally repairable codes that satisfy some further properties have been considered in recent works, such as \([19], [27], [31]\). The model of adversarial node failure is adapted universally almost everywhere without any consideration for the statistics of server failures. This results in a combinatorial simplification, but the designed codes may be far from being the best. On the other hand, just as in the case of update-efficiency, it is reasonable to assume that the servers fails independently (and with equal probability) with small failure probability. However, so far we have not seen any work that considers such capacity results for locally recoverable codes. We attempt to address this gap, at least in part.

We show (and relatively easily) that it is possible to construct codes with rate \( \epsilon \) less than the capacity of the BSC (or the BEC) that have local recoverability \( O(\log 1/\epsilon) \), and simultaneously update-efficiency scaling logarithmically with the block-length. Our main result in this area is a converse result that shows that the scaling \( O(\log 1/\epsilon) \) is optimal for a BEC. Indeed, if the rate of a code that achieves arbitrarily small probability of error over a BEC is \( \epsilon \) less than the capacity then the local recoverability is \( \Omega(\log 1/\epsilon) \).

C. Channel models: BEC vs. BSC

Most of the results of this paper concern the binary erasure channel with erasure probability \( p \), i.e., \( \text{BEC}(p) \), and extends naturally, or with some effort, to the case of binary symmetric channel with flip probability \( p \), i.e., \( \text{BSC}(p) \). The capacity of \( \text{BSC}(p) \) is \( 1 - h_B(p) \), \( h_B(p) = -p \log_2(p) - (1 - p) \log_2(1 - p) \) being the binary entropy function, and the capacity of \( \text{BEC}(p) \) is
1 − p. It should be noted that erasure channel model is often a more natural setting in distributed storage systems.

In particular, the result of Sec. III, the existence of capacity achieving simultaneously locally recoverable and update-efficient codes, holds true for both BEC and BSC. The converse result regarding update-efficient linear codes (section IV-A) holds for both BSC and BEC, while the converse for general code (Sec. IV-B) applies to only BSC. The converse result for local recovery (Sec. V) is for only the BEC.

D. Larger alphabet: binary to $q$-ary

Although our results are derived for binary-input channels, as opposed to the large alphabet channel models usually considered for distributed storage, our proofs extend to large alphabet case. The $q$-ary generalizations for BSC and BEC are respectively the $q$-ary symmetric channel and $q$-ary erasure channel. The definitions and capacities of these channels are standard and can be found in textbooks, for example, in [28, §1.2 & §1.5.3].

The existential result of Sec. III extends to the case of $q$-ary channels. See, [3, §IIIB, remark 6] for more detail on how the error-exponent result for BSC extends to $q$-ary case. There are also results concerning the error-exponent of $q$-ary low density codes that can be used to extend Theorem 5. The result one can most directly use is perhaps [12].

The converse results for Sec. IV and V in particular Theorem 4, Theorem 5 and Theorem 12 can easily be stated for the case of $q$-ary channel. The observations regarding adversarial error case of Sec. VI-A is also extendable to $q$-ary case in a straightforward manner.

E. Organization

The organization of the paper is as follows. Section II introduces the notation that will be used throughout the paper. In Section III we show that there exist linear codes of length $n$ and rate $\epsilon$ less than capacity, with update-efficiency logarithmic in blocklength and local recoverability $O(\log 1/\epsilon)$. In Section IV our main impossibility results for capacity-achieving update-efficient codes are presented. Subsequently in Section V we address the local recoverability of capacity-achieving codes and deduce a converse result that matches the achievability part. In Section VI,
we make a digression and discuss the performance of update-efficient codes for the adversarial failure model. We provide examples of optimal update-efficient codes, and discuss the generalized notion of update-efficiency. We note that the notions of update-efficiency and local recovery are also applicable to source coding or data-compression codes. In the last section, Section VII, of this paper, we briefly discuss the dual problem of lossy source coding in the context of update-efficiency and local recoverability and conclude with some remarks.

II. DEFINITIONS AND NOTATION

In this paper, a code $C \in \mathbb{F}_2^n$ is a collection of binary $n$-vectors. The support of a vector $x$ (written as $\text{supp}(x)$) is the set of coordinates where $x$ has nonzero values. By the weight of a vector we mean the size of support of the vector. It is denoted as $\text{wt}(\cdot)$. The logarithms in this paper have base 2 unless otherwise mentioned.

Let $\mathcal{M}$ be the set of all possible messages. Usually accompanied with the definition of the code, is an injective encoding map $\phi: \mathcal{M} \rightarrow C$, which defines how the messages are mapped to codewords. In the following discussion, let us assume $\mathcal{M} = \mathbb{F}_2^k$. In an update-efficient code, for all $x \in \mathbb{F}_2^k$, and for all $e \in \mathbb{F}_2^k : \text{wt}(e) \leq u$, we have $\phi(x + e) = \phi(x) + e'$, for some $e' \in \mathbb{F}_2^n : \text{wt}(e') \leq t$. A special case of this is captured in the following definition.

Definition 1: The update-efficiency of a code $C$ and the encoding $\phi$, is the maximum number of bits that needs to be changed in a codeword when 1 bit in the message is changed. A code $(C, \phi)$ has update-efficiency $t$ if for all $x \in \mathbb{F}_2^k$, and for all $e \in \mathbb{F}_2^k : \text{wt}(e) = 1$, we have $\phi(x + e) = \phi(x) + e'$, for some $e' \in \mathbb{F}_2^n : \text{wt}(e') \leq t$.

A linear code $C \in \mathbb{F}_2^n$ of dimension $k$ is a $k$-dimensional subspace of the vector space $\mathbb{F}_2^n$. For linear codes the mapping $\phi: \mathbb{F}_2^k \rightarrow C$ is naturally given by a set of bases of the code: if we arrange a set of bases of the code as rows of a matrix $G$, then $\phi(x) = x^T G$, for any $x \in \mathbb{F}_2^k$. This $k \times n$ matrix $G$ is called the generator matrix. There can be a number of generator matrices for a code $C$, which correspond to different labelings of the codewords. By an $[n, k, d]$ code we mean a linear code with length $n$, dimension $k$ and minimum (pairwise) distance between the codewords $d$. Linear codes form the most studied classes of error-correcting codes, and have a
number of benefits in terms of representation and encoding and decoding complexity.

For a linear code, when changing one bit in the message, the maximum number of bits that need to be changed in the codeword is the maximum over the weights of the rows of the generator matrix. Hence, for an update-efficient code, we need a representation of the linear code where the maximum weight of the rows of the generator matrix is low.

Proposition 1: A linear code $C$ will have update-efficiency $t$ if and only if there is a generator matrix $G$ of $C$ with maximum row weight $t$.

Proof: It is easy to see that if the maximum number of ones in any row is bounded above by $t$, then at most $t$ bits need to be changed to update one bit change in the message.

On the other hand if the code has update-efficiency $t$ then there must exist a labeling $\phi$ that gives a sparse generator matrix. Specifically the vectors $(1,0,\ldots,0), (0,1,\ldots,0), \ldots, (0,0,\ldots,1) \in \mathbb{F}_2^k$ must produce vectors of weight at most $t$ under $\phi$. Therefore, the generator matrix given by $\phi$ will have row weight at most $t$.

This implies, given a linear code, to see whether it is update-efficient or not, we need to find the sparsest basis for the code. A linear code with a sparse basis is informally called a low density generator matrix (LDGM) code.

There are a number of different ways that local recovery could be defined. The simplest is perhaps the one given below, which insists that for each codeword symbol, there is a set of at most $r$ codeword positions that need to be queried to recover the given symbol with certainty. A weaker definition could allow adaptive queries, i.e., the choice of which $r$ positions to query could depend on the values of previously queried symbols. Finally, one could ask that instead of obtaining the value of the codeword symbol with certainty, one obtains the value with some probability significantly higher than .5. For simplicity, we sketch all the arguments in this paper for the simplest definition, i.e., Defn. 2. The arguments can easily be extended to the other definitions, except for some cases that will be explicitly mentioned later.

Definition 2: A code $C \subseteq \mathbb{F}_2^n$ has local recoverability $r$, if for any $x = (x_1, \ldots, x_n) \in C$ and for any $1 \leq i \leq n$, there exists a function $f_i : \mathbb{F}_2^r \rightarrow \mathbb{F}_2$ and indices $1 \leq i_1, \ldots, i_r \leq n, i_j \neq i, 1 \leq j \leq r$, such that $x_i = f_i(x_{i_1}, \ldots, x_{i_r})$. 
A generator matrix $H$ of the null-space of a linear code $C$ is called a parity-check matrix for $C$. It is to be noted that for any $x \in C$, $Hx = 0$. A low density parity-check (LDPC) code is a linear code with a parity check matrix such that each row of the parity check matrix has a small (constant) number of nonzero values. The following proposition is immediate.

**Proposition 2:** If the maximum row-weight of a parity-check matrix of a code is $r$, then the code has local recoverability at most $r$.

Hence, LDPC codes are locally recoverable, although this is not a necessary condition of local recoverability. Note the following necessary condition for local recovery with linear codes: the union of supports of all low-weight rows of parity-check matrix must include all the $n$ coordinates. On the other hand in LDPC codes every row is low-weight.

### III. Existence of Good Codes

In this section, our aim is to show, in a rather simple way, that there exist linear codes of length $n$ that

1) have rate $\epsilon$ less than capacity, $\epsilon > 0$,
2) achieve arbitrarily small probability of error,
3) have update-efficiency $O(\log n)$ and
4) have local recoverability $O(\log 1/\epsilon)$.

It is relatively easy to construct a code with local recoverability $O(\log 1/\epsilon)$ that achieves capacity over the BSC or BEC within a additive term $\epsilon$. One can in principle choose the rows of the parity-check matrix randomly from all low weight vectors, and argue that this random ensemble contain many codes that achieve the capacity of the binary symmetric channel (BSC) up to an additive term $\epsilon$. Indeed, LDPC codes achieve the capacity of the binary symmetric channel [10].

Similarly, one may try to construct a low row-weight generator matrix randomly to show that the ensemble average performance achieves capacity. In this direction, some steps have been taken in [18]. However, these constructions fails to achieve local recoverability and update-efficiency simultaneously. Below, we describe one construction that indeed does that.
It is known that for every $\epsilon > 0$ and any sufficiently large $n$, there exist a linear code of length $n$ and rate $1 - h_B(p) - \epsilon$ that has probability of incorrect decoding at most $2^{-E(p,\epsilon)n}$. There are numerous evaluations of this result and estimates of $E(p,\epsilon) > 0$. We refer the reader to [3] as an example.

Let $m = (1 + \alpha)/E(p,\epsilon) \log n$, an integer, $\epsilon, \alpha > 0$. We avoid using ceiling and floor to have a clean presentation, unless it is not obvious from the context. We know that for sufficiently large $n$, there exists a linear code $\hat{C}$ given by the $mR \times m$ generator matrix $\hat{G}$ with rate $R = 1 - h_B(p) - \epsilon$ that has probability of incorrect decoding at most $2^{-E(p,\epsilon)m}$.

Let $G$ be the $nR \times n$ matrix that is the Kronecker product of $\hat{G}$ and the $n/m \times n/m$ identity matrix $I_{n/m}$, i.e.,

$$G = I_{n/m} \otimes \hat{G}.$$  

Clearly a codeword of the code $\mathcal{C}$ given by $G$ is given by $n/m$ codewords of the code $\hat{C}$ concatenated side-by-side. The probability of error of $\mathcal{C}$ is therefore, by the union bound, at most

$$\frac{n}{m} 2^{-E(p,\epsilon)m} = \frac{nE(p,\epsilon)}{(1 + \alpha)n^{1+\alpha} \log n} = \frac{E(p,\epsilon)}{(1 + \alpha)n^{\alpha} \log n}.$$  

However, notice that the generator matrix has row weight bounded above by $m = (1 + \alpha)/E(p,\epsilon) \log n$. Hence, we have constructed a code with update-efficiency $(1 + \alpha)/E(p,\epsilon) \log n$, and rate $1 - h_B(p) - \epsilon$ that achieves a probability of error less than $E(p,\epsilon)/[(1 + \alpha)n^{\alpha} \log n]$ over a BSC$(p)$.

We modify the above construction slightly to produce codes that also possess good local recoverability. It is known that LDPC codes achieve a positive error-exponent. That is, for every $\epsilon > 0$ and any sufficiently large $n$, there exist an LDPC code of length $n$ and rate $1 - h_B(p) - \epsilon$ that has check degree (number of 1s in a row of the parity-check matrix) at most $O(\log 1/\epsilon)$, and probability of incorrect decoding at most $2^{-E_L(p,\epsilon)n}$, for some $E_L(p,\epsilon) > 0$. We refer the reader to [10], [22] for more details of this result. This code will be chosen as $\hat{C}$ in the above construction, and $\hat{G}$ can be any generator matrix for $\hat{C}$.

The construction now follows without any more changes. We have, $m = (1 + \alpha)/E_L(p,\epsilon) \log n$, an integer, $\epsilon, \alpha > 0$, and $G = I_{n/m} \otimes \hat{G}$. 

Now, the generator matrix has row weight bounded above by $m = (1 + \alpha)/E_L(p, \epsilon) \log n$. So, the code has update-efficiency $(1 + \alpha)/E_L(p, \epsilon) \log n$, rate $1 - h_B(p) - \epsilon$, and achieves probability of error less than $E_L(p, \epsilon)/[(1 + \alpha)n^\alpha \log n]$ over a BSC($p$).

Moreover, the parity-check matrix of the resulting code will be block-diagonal, with each block being the parity-check matrix of the code $\hat{C}$. The parity-check matrix of the overall code has row-weight $O(\log 1/\epsilon)$. Hence, any codeword symbol can be recovered from at most $O(\log 1/\epsilon)$ other symbols by solving one linear equation. Therefore, we have the following result.

**Theorem 3:** There exists a family of linear codes $C_n$ of length $n$ and rate $1 - h_B(p) - \epsilon$, that have probability of error over BSC($p$) going to 0 as $n \rightarrow \infty$. These codes simultaneously achieve update-efficiency $O(\log n/E_L(p, \epsilon))$ and local recoverability $O(\log 1/\epsilon)$.

Hence, it is possible to simultaneously achieve local recovery and update-efficiency with a capacity-achieving code on BSC($p$). A similar result is immediate for BEC($p$).

**IV. IMPOSSIBILITY RESULTS FOR UPDATE-EFFICIENCY**

In this section, we show that for suitably small $\alpha$, no code can simultaneously achieve capacity and have update-efficiency better than $\alpha \log n$, $n$ blocklength. More precisely, we give the following converse results.

1) **Linear codes.** Linear codes of positive rate cannot have arbitrarily small probability of error and update-efficiency better than $\alpha \log n, \alpha > 0$ when used over the BEC. Since a BSC is degraded with respect to a BEC, this result implies same claim for BSC as well. To see that BSC($p$) is a degraded version of a BEC with erasure probability $2p$, one can just concatenate BEC($2p$) with a channel with ternary input $\{0, 1, ?\}$ and binary output $\{0, 1\}$, such that with probability 1 the inputs $\{0, 1\}$ remain the same, and with uniform probability $?$ goes to $\{0, 1\}$.

2) **General codes.** Any (possibly non-linear) code with positive rate cannot have update-efficiency better than $\alpha \log n, \alpha > 0$, and vanishing probability of error when transmitted over BSC. The value of $\alpha$ that we obtain in this case is larger than that in the case of linear codes; moreover this result applies to more general codes than the previous. But
we have not been able to extend it to the BEC. It could be interesting to explore whether nonlinear codes of positive rate must have at least logarithmic update efficiency for the BEC.

3) *LDGM ensemble.* We also show that for the ensemble of LDGM codes with fixed row-weight $\alpha \log n$, $\alpha > 0$, almost all codes have probability of error $\sim 1$ when transmitted over a BSC. The value of $\alpha$ in this case is much larger than the previous two cases.

A plot providing the lower bound on update-efficiency of “good” codes is presented in Fig. 1. In this figure, the values of $\alpha$, the constant multiplier of $\ln n$, as a function of BSC flip probability $p$ is plotted. The plot contains results of Theorems 4, 5 and 7. Note that $\alpha(p) \to \infty$ as $p \to 1/2$ for general codes (Theorem 5) and the LDGM ensemble (Theorem 7).

### A. Impossibility result for linear codes

The converse result for linear codes used over a binary erasure channel is based on the observation that when the update-efficiency is low, the generator matrix $G$ is very sparse, i.e., every row of $G$ has very few non-zero entries. Let the random subset $I \in \{1, \ldots, n\}$ denote the coordinates not erased by the binary erasure channel. Let $G_I$ denote the submatrix of $G$ induced by the unerased received symbols, i.e., the columns of $G$ corresponding to $I$. Then, because $G$ is so sparse, it is quite likely that $G_I$ has several all zero rows, and the presence of such rows implies a large error probability. We formalize the argument below.

**Theorem 4:** Consider using some linear code of length $n$, dimension $k$ and update-efficiency $t$, specified by generator matrix $G$ over BEC($p$). Hence, all rows of $G$ has weight at most $t$. Assume that for some $\epsilon > 0,$

$$t < \frac{\ln \frac{k^2}{2n \ln(1/\epsilon)}}{2 \ln \frac{1}{p}}.$$  

Then, the average probability of error is at least $1/2 - \epsilon$.

**Proof:** For linear codes over the binary erasure channel, analyzing the probability of error essentially reduces to analyzing the probability that the matrix $G_I$ induced by the unerased columns of $G$ has rank $k$ (note that the rank is computed over $\mathbb{F}_2$). To show that the rank is
likely to be less than $k$ for sufficiently small $t$, let us first compute the expected number of all zero rows of $G_I$. Since every row of $G$ has weight at most $t$, the expected number of all zero rows of $G_I$ is at least $kp^t$. The rank of $G_I$, $\text{rank}(G_I)$, is at most $k$ minus the number of all zero rows, so the expected rank of $G_I$ is at most $k - kp^t$.

Now, observe that the rank is a 1-Lipschitz functional of the independent random variables denoting the erasures introduced by the channel. Therefore, by Azuma’s inequality [1, Theorem 7.4.2], the rank of $G_I$ satisfies

$$\Pr(\text{rank}(G_I) \geq \mathbb{E}\text{rank}(G_I) + \lambda) < e^{-\frac{\lambda^2}{2n}}.$$
Therefore,
\[
\Pr(\text{rank}(G_I) \geq k - kp' + \lambda) < e^{-\frac{\lambda^2}{2n}}.
\]

In particular,
\[
\Pr(\text{rank}(G_I) = k) < e^{-\frac{k^2p^2t}{4n}}.
\]

Assuming the value given for \( t \), we see that
\[
\Pr(\text{rank}(G_I) = k) < \epsilon.
\]

Since even the maximum likelihood decoder makes an error with probability at least 0.5 when \( \text{rank}(G_I) < k \), this shows that when
\[
t < \frac{\ln \frac{k^2}{2n \ln(1/\epsilon)}}{2 \ln \frac{1}{p}},
\]
the probability of error is at least \( 1/2 - \epsilon \). (In fact, the average error probability converges to 1. The above argument can easily be extended to show that the probability of decoding successfully is at most \( e^{-\Omega(k^4/\log k)} \) for some \( \delta > 0 \), but we omit the details.)

\[\blacksquare\]

\textbf{B. Impossibility for general codes}

Now, we prove that even nonlinear codes cannot have low update-efficiency for the binary symmetric channel. The argument is based on a simple observation. If a code has dimension \( k \) and update-efficiency \( t \), then any given codeword has \( k \) neighboring codewords within distance \( t \), corresponding to the \( k \) possible 1-bit changes to the information bits. If \( t \) is sufficiently small, it is not possible to pack \( k + 1 \) codewords into a Hamming ball of radius \( t \) and maintain a low probability of error.

\textit{Theorem 5:} Consider using some (possibly non-linear) code of length \( n \), dimension (possibly fractional) \( k \) and update-efficiency \( t \) over BSC(\( p \)). Assume that \( t \leq (1 - \alpha) \log k / \log((1 - p)/p) \), for some \( \alpha > 0 \). Then, the average probability of error is at least \( 1 - o(1) \), where \( o(1) \) denotes a quantity that goes to zero as \( k \to \infty \).

\textit{Proof:} First, we show that a code consisting of \( k + 1 \) codewords contained in a Hamming ball of radius \( t \) has large probability of error. Instead of analyzing BSC(\( p \)), consider the closely related
channel where exactly \(w\) uniformly random errors are introduced. For this channel, subject to the constraint that the \(k+1\) codewords are contained in a Hamming ball of radius \(t\), the average probability of error is at least
\[
1 - \frac{(2t + 1) \binom{n}{w+t}}{(k+1) \binom{n}{w}} \geq 1 - \frac{2t(n-w)^t}{kw^t}.
\]
To see this, take \(x_1, \ldots, x_{k+1}\) to be the codewords, and \(B_i, i = 1, \ldots, k+1\), to be the corresponding decoding regions. Without loss of generality, we can assume that the ML decoder is deterministic, so the \(B_i\)'s are all disjoint. Now, let \(D_i\) be the set of possible outputs of the channel for input \(x_i, i = 1, \ldots, k+1\). The average probability of correct decoding is
\[
\frac{1}{k+1} \sum_{i=1}^{k+1} \frac{|B_i \cap D_i|}{\binom{n}{w}} = \frac{1}{k+1} \frac{\left| \bigcup_i (B_i \cap D_i) \right|}{\binom{n}{w}}.
\]
But \(\bigcup_i D_i \leq \sum_{j=1}^{t} \binom{n}{w+j} \leq (2t+1) \binom{n}{w+t} \).

(Note: throughout the proof, we assume that \(w+t < n/2\).) Now, the binary symmetric channel will introduce at least \(pn - n^{2/3}\) errors with probability at least \(1 - o(1)\), and these errors are uniformly distributed. Therefore, the probability of error on the binary symmetric channel is at least
\[
1 - \frac{2t(1-p)^t}{kp^t} + o(1).
\]
If \(t \leq (1-\alpha) \log k / \log((1-p)/p)\), this implies that the probability of error is greater than \(1 - 2t/k^\alpha + o(1)\).

Now, for each message \(x\) of the given \((n,k)\) code with update-efficiency \(t\), consider the subcode \(C_x\) consisting of the \(k+1\) codewords \(\phi(x), \phi(x+e_1), \ldots, \phi(x+e_k)\), corresponding to the encodings of \(x\) and the \(k\) messages obtained by changing a single bit of \(x\). These codewords lie within a Hamming ball of radius \(t\) centered around \(\phi(x)\). The above argument shows that even a maximum likelihood decoder has a large average probability of error for decoding the subcode \(C_x\). Let us call this probability \(P_{C_x}\). We claim that the average probability of error of the code \(C\) with maximum likelihood decoding, \(P_C\), is at least the average, over all \(x\), of the probability of error for the code \(C_x\), up to some factor. In particular,
\[
P_C \geq \frac{k}{n |C|} \sum_{x \in C} P_{C_x}.
\]
We will now prove this claim and thus the theorem. Note that \( P_e = 1/|\mathcal{C}| \sum_{x \in \mathcal{C}} P_x \), where \( P_x \) is the probability of error if codeword \( x \) is transmitted. Therefore,

\[
P_{e_x} = \frac{1}{|\mathcal{C}_x|} \sum_{y \in \mathcal{C}_x} P_y.
\]

We have,

\[
\frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} P_{e_x} \leq \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} \frac{1}{|\mathcal{C}_x|} \sum_{y \in \mathcal{C}_x} P_y = \frac{1}{(k+1)|\mathcal{C}|} \sum_{x \in \mathcal{C}} \sum_{y \in \mathcal{C}_x} P_y = \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} \frac{d_x}{k+1} P_x,
\]

where \( d_x = |\{ y : x \in \mathcal{C}_y \}| \leq n \). Hence,

\[
\frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} P_{e_x} \leq \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} \frac{n}{k+1} P_x = \frac{n}{k+1} P_e.
\]

We conclude that the original code \( \mathcal{C} \) has probability of error at least \( 1 - o(1) \) when

\[
t \leq \frac{(1 - \alpha) \log k}{\log(\frac{k}{p})}.
\]

Remark 1: This argument does not work for the binary erasure channel. In fact, there exist zero rate codes for the binary erasure channel with vanishing error probability and sub-logarithmic update-efficiency. Specifically, consider an encoding from \( k \) bits to \( 2^k \) bits that maps a message \( x \) to the string consisting of all 0’s except for a single 1 in the position with binary expansion \( x \). Repeat every symbol of this string \( c \) times to obtain the final encoding \( \phi(x) \). The update-efficiency is \( 2c \), since every codeword has exactly \( c \) 1’s, and different codewords never have a nonzero entry in the same position. Since the location of a nonzero symbol uniquely identifies the message, the error probability is at most the probability that all \( c \) 1’s in the transmitted codeword are erased, i.e., at most \( p^c \). Therefore, we achieve vanishing error probability as long as \( c \rightarrow \infty \), and \( c \) can grow arbitrarily slowly.

We conjecture that for positive rates, even nonlinear codes must have logarithmic update complexity for the binary erasure channel.
C. Ensemble of LDGM codes

Let us motivate the study of one particular ensemble of LDGM codes here. Suppose we want to construct a code with update-efficiency \( t \). From proposition 1, we know that a linear code with update-efficiency \( t \) always has a generator matrix with maximum row weight \( t \). For simplicity we consider generator matrices with all rows having weight exactly \( t \). We look at the ensemble of linear codes with such generator matrices, and show that almost all codes in this ensemble are bad for \( t \) less than certain value. Note that any \( k \times n \) generator matrix with row weight at most \( t \) can be extended to a generator matrix with block-length \( n + t - 1 \) and row weight exactly \( t \) (by simply padding necessary bits in the last \( t - 1 \) columns).

Let \( \Gamma_{n,k,t} \) be the set of all \( k \times n \) matrices over \( \mathbb{F}_2 \) such that each row has exactly \( t \) ones. First of all, we claim that almost all the matrices in \( \Gamma_{n,k,t} \) generate codes with dimension \( k \) (i.e., the rank of the matrix is \( k \)). Indeed, we quote the following lemma from [6].

**Lemma 6:** Randomly and uniformly choose a matrix \( G \) from \( \Gamma_{n,k,t} \). If \( k \leq (1 - e^{-t} / \ln 2 - o(e^{-t})) n \), then with probability \( 1 - o(1) \) the rank of \( G \) is \( k \).

This lemma, along with the next theorem, which is the main result of this section, will show the fact claimed at the start of this section.

**Theorem 7:** Fix an \( 0 < \alpha < 1/2 \). For at least a \( 1 - t^2 n^{2\alpha} / (n - t) \) proportion of the matrices in \( \Gamma_{n,k,t}, k \geq n^\alpha \), the corresponding linear code has probability of error at least \( n^\alpha 2^{-\lambda_p t} / \sqrt{t} \) over a BSC(\( p \)), for \( p < 1/2 \) and \( \lambda_p = -1 - 1/2 \log p - 1/2 \log(1 - p) > 0 \).

The proof of this theorem is deferred until later in this section. This theorem implies that for any \( \alpha < 1/2 \), most codes in the random ensemble of codes with fixed row-weight (and hence update-efficiency) \( t < \alpha / \lambda_p \log n \) have probability of error bounded away from 0 for any positive rate. Indeed, we have the following corollary.

**Corollary 8:** For at least \( 1 - o(1) \) proportion of all linear codes with fixed \( t \)-row-weight generator matrix, \( t < (\alpha - \epsilon) / \lambda_p \log n \), \( \alpha < 1/2 \), \( \epsilon > 0 \), and dimension \( k, k > n^\alpha \), the probability of error is \( 1 - o(1) \) over a BSC(\( p \)), for \( 0 < p \leq 1/2 \).

In particular, this shows that codes with fixed row weight \( t < 1/(2\lambda_p) \log n \) and rate greater than \( 1/n^{1-\alpha} \) are almost always bad.
Proof of Corollary 8: From Lemma 6 it is clear that a $1-o(1)$ proportion of all codes in $\Gamma_{n,k,t}$ have rank $k$. Hence, if a $1-o(1)$ proportion of codes in $\Gamma_{n,k,t}$ have some property, a $1-o(1)$ proportion of codes with $t$-row-weight generator matrix and dimension $k$ also have that property.

Now, plugging in the value of $t$ in the expression for probability of error in Theorem 7, we obtain the corollary.

To prove Theorem 7 we will need the following series of lemmas.

Lemma 9: Let $x \in \{0,1\}^n$ be a vector of weight $t$. Let the all-zero vector of length $n$ be transmitted over a BSC with flip probability $p < 1/2$. If the received vector is $y$, then

$$\Pr(\text{wt}(y) > d_H(x, y)) \geq \frac{1}{\sqrt{t}}2^{-\lambda_p t},$$

where $\lambda_p = -1 - 1/2 \log p - 1/2 \log(1-p) > 0$.

Proof: Let $I \subset [n]$ be the support of $x$. We have $|I| = t$. Now, $\text{wt}(y) > d_H(x, y)$ whenever the number of errors introduced by the BSC in the coordinates $I$ is $> t/2$. Hence,

$$\Pr(\text{wt}(y) > d_H(x, y)) = \sum_{i> t/2} \binom{t}{i} p^i (1-p)^{t-i}$$

$$> \binom{t}{t/2} p^{t/2} (1-p)^{t-t/2} \geq \frac{1}{\sqrt{t}}2^{-\lambda_p t}.$$ 

Lemma 10: Suppose two random vectors $x, y \in \{0,1\}^n$ are chosen independently and uniformly from the set of all length-$n$ binary vectors of weight $t$. Then,

$$\Pr(\text{supp}(x) \cap \text{supp}(y) = \emptyset) > 1 - \frac{t^2}{n-t+1}.$$ 

Proof: The probability in question equals

$$\frac{\binom{n-t}{t}}{\binom{n}{t}} = \frac{(n-t)!^2}{(n-2t)!n!}$$

$$= \frac{(n-t)(n-t-1)(n-t-2) \ldots (n-2t+1)}{n(n-1)(n-2) \ldots (n-t+1)}$$

$$= \left(1 - \frac{t}{n}\right) \left(1 - \frac{t}{n-1}\right) \ldots \left(1 - \frac{t}{n-t+1}\right)$$
\[
> \left(1 - \frac{t}{n-t+1}\right)^t \geq 1 - \frac{t^2}{n-t+1}.
\]

In the last step we have truncated the series expansion of \(\left(1 - \frac{t}{n-t+1}\right)^t\) after the first two terms. The inequality will be justified if the terms of the series are decreasing in absolute value. Let us verify that to conclude the proof. In the following \(X_i\) denote the \(i\)th term in the series, \(0 \leq i \leq t\).

\[
\frac{X_{i+1}}{X_i} = \frac{\binom{t}{i+1}}{t} \cdot \frac{t}{n-t+1} = \frac{t-i}{i+1} \cdot \frac{t}{n-t+1} \leq 1,
\]

for all \(i \leq t-1\).

\textbf{Lemma 11:} Let us choose any \(n^\alpha, 0 < \alpha < 1/2\), random vectors of weight \(t\) independently and uniformly from the set of weight-\(t\) vectors. Denote the vectors by \(\mathbf{x}_i, 1 \leq i \leq n^\alpha\). Then,

\[
\Pr(\forall i \neq j, \text{supp}(\mathbf{x}_j) \cap \text{supp}(\mathbf{x}_i) = \emptyset) \geq 1 - \frac{t^2 n^{2\alpha}}{n-t}.
\]

This implies all of the vectors have disjoint supports with probability at least \(1 - t^2 n^{2\alpha}/(n-t)\).

\textbf{Proof:} The claim follows by taking a union bound over all pairs of randomly chosen vectors.

Now, we are ready to prove Theorem 7.

\textbf{Proof of Theorem 7:} We begin by choosing a matrix \(G\) uniformly at random from \(\Gamma_{n,k,t}\). This is equivalent of choosing each row of \(G\) uniformly and independently from the set of all \(n\)-length \(t\)-weight binary vectors. Now, \(k > n^\alpha\), hence there exists \(n^\alpha\) vectors among the rows of \(G\) such that any two of them have disjoint support with probability at least \(1 - t^2 n^{2\alpha}/(n-t)\) (from Lemma 11). Hence, for at least a proportion \(1 - t^2 n^{2\alpha}/(n-t)\) of matrices of \(\Gamma_{n,k,t}\), there are \(n^\alpha\) rows with disjoint supports. Suppose \(G\) is one such matrix. It remains to show that the code \(\mathcal{C}\) defined by \(G\) has probability of error at least \(n^\alpha 2^{-\lambda p t}/\sqrt{t}\) over BSC\((p)\).

Suppose, without loss of generality, that the all zero vector is transmitted over a BSC\((p)\), and \(\mathbf{y}\) is the vector received. We know that there exists at least \(n^\alpha\) codewords of weight \(t\) such that all of them have disjoint support. Let \(\mathbf{x}_i, 1 \leq i \leq n^\alpha\), be those codewords. Then, the probability that the maximum likelihood decoder incorrectly decodes \(\mathbf{y}\) to \(\mathbf{x}_i\) is

\[
\Pr(\text{wt}(\mathbf{y}) > d_H(\mathbf{x}_i, \mathbf{y})) \leq \frac{1}{\sqrt{t}} 2^{-\lambda p t}.
\]
from Lemma 9. As the codewords $\mathbf{x}_1, \ldots, \mathbf{x}_{n^\alpha}$ have disjoint supports, the probability that the maximum likelihood decoder incorrectly decodes to any one of them is at least

$$1 - \left(1 - \frac{1}{\sqrt{t}} 2^{-\lambda pt}\right)^{n^\alpha} = (1 - o(1)) \cdot \frac{n^\alpha}{\sqrt{t}} 2^{-\lambda pt}.$$  

Remark 2: Theorem 7 is also true for the random ensemble of matrices where the entries are independently chosen from $\mathbb{F}_2$ with $\Pr(1) = \frac{t}{n}$.

V. IMPOSSIBILITY RESULT FOR LOCAL RECOVERY

Now that we have our main impossibility result on update-efficiency we turn to the local recovery property. In this section, we deduce the converse result concerning local recovery for the binary erasure channel. We show that any code with a given local recoverability has to have rate bounded away from capacity to provide arbitrarily small probability of error, when used over the binary erasure channel. In particular, for any code, including non-linear codes, recovery complexity at a gap of $\epsilon$ to capacity on the BEC must be at least $\Omega(\log 1/\epsilon)$, proving that the above LDPC construction is simultaneously optimal to within constant factors for both update-efficiency and local recovery.

The intuition for the converse is that if a code has low local recovery complexity, then codeword positions can be predicted by looking at a few codeword symbols. As we will see, this implies that the code rate must be bounded away from capacity, or the probability of error approaches 1. In a little more detail, for an erasure channel, the average error probability is related to how the codewords behave under projection onto the unerased received symbols. Generally, different codewords may result in the same string under projection, and without loss of generality, the ML decoder can be assumed to choose a codeword from the set of codewords matching the received channel output in the projected coordinates uniformly at random. Thus, given a particular erasure pattern induced by the channel, the average probability of decoding success for the ML decoder is simply the number of different codeword projections, divided by $2^{Rn}$, the size of the codebook. We now show that the number of different projections is likely to be far less than $2^{Rn}$.
Theorem 12: Let $C$ be a code of length $n$ and rate $1 - p - \epsilon$ that achieves probability of error less than 0.5 when used over BEC($p$). Then, the local recoverability of $C$ is at least $c \log 1/\epsilon$, for some constant $c > 0$.

Proof: Let $C$ be a code of length $n$ and size $2^{nR}$ that has local recoverability $r$. Let $T$ be the set of coordinates with the property that the query positions required to recover these coordinates appear before them. To show that such an ordering exists with $|T| \geq n/(r + 1)$, we can randomly and uniformly permute the coordinates of $C$. The expected number of such coordinates is then $n/(r + 1)$, hence some ordering exists with $|T| \geq n/(r + 1)$.

Assume $I \subseteq \{1, \ldots, n\}$ is the set of coordinates erased by the BEC, and let $\tilde{I} = \{1, \ldots, n\} \setminus I$. Let $x \in C$ be a randomly and uniformly chosen codeword. $x_I$ and $x_{\tilde{I}}$ denote the projection of $x$ on the respective coordinates. We are interested in the logarithm of the number of different codeword projections onto $\tilde{I}$, which we denote by $\log S(x_{\tilde{I}})$. Note that this is a random-variable with respect to the random choice of $I$ by the BEC.

Suppose that the number of elements of $T$ that have all $r$ of their recovery positions un-erased is $u$. Then, the number of different codeword projections is unchanged if we remove these $u$ elements from $T$. Hence,

$$\log S(x_{\tilde{I}}) \leq |\tilde{I}| - u.$$  

But $\mathbb{E}u \geq (1 - p)^r |T|$. Therefore,

$$\mathbb{E} \log S(x_{\tilde{I}}) \leq n(1 - p) - (1 - p)^r \frac{n}{r + 1}.$$  

Observe that $\log S(x_{\tilde{I}})$ is a 1-Lipschitz functional of independent random variables (erasures introduced by the channel). This is because projecting onto one more position cannot decrease the number of different codeword projections, and at most doubles the number of projections. Therefore, we can use Azuma’s inequality to conclude that

$$\Pr \left( \log S(x_{\tilde{I}}) > n(1 - p) - (1 - p)^r \frac{n}{r + 1} + \alpha \right) \leq e^{-\frac{\alpha^2}{2n}}.$$  

If we set

$$r = \log \frac{1}{[(r + 1)(2\epsilon + \alpha)]} \log \frac{1}{(1 - p)},$$  

then
then
\[
\Pr \left( \log S(x_f) > n(1 - p - 2\epsilon) \right) \leq e^{-\frac{n^2}{2}}.
\]

This means that for a suitable constant \( c \), if \( r \leq c \log 1/\epsilon \), then with very high probability \( \log S(x_f) \leq n(1 - p - 2\epsilon) \). However, there are \( 2^{Rn} = 2^{n(1-p-\epsilon)} \) codewords, so we conclude that the probability of successful decoding is at most
\[
2^{-cn} + e^{-\frac{n^2}{2}}.
\]

Thus, we have proved that if \( r \leq c \log 1/\epsilon \), the probability of error converges to 1, and in particular, is larger than .5 for sufficiently large \( n \).

\[ \text{Remark 3:} \] Rather than considering the number of different codeword projections, we could have considered the entropy of the distribution of codeword projections onto \( I \), which is also a 1-Lipschitz functional. This is a more general approach that can be extended to the case where local recovery can be adaptive and randomized, and only has to succeed with a certain probability (larger than .5), as opposed to providing guaranteed recovery. However, one obtains a bound of \( n(1 - p - 2\epsilon) \) on the entropy, so Fano’s inequality only shows that the probability of error must be \( \Omega(\epsilon) \), while the above analysis shows that the probability of error must be close to 1.

\[ \text{VI. MORE ON UPDATE-EFFICIENCY} \]

In this section we discuss some further observations regarding the update-efficiency of codes. We first make a digression from the capacity results for BSC and BEC to a channel model ubiquitously studied in the storage literature: the adversarial error model. We then go on to extend the definition of update-efficiency to a more general setting.

\[ \text{A. Adversarial channels} \]

In an adversarial error model, the channel is allowed to introduce any up to \( s \) errors (or \( 2s \) erasures). It is known that to correct \( s \) errors (\( 2s \) erasures), the minimum distance of the code needs to be at least \( 2s + 1 \). However, if a code has update-efficiency \( t \), then there must
exist two (in fact, many more) codewords that are within distance $t$ of each other. Hence, small update-efficiency implies less error correction capability.

In particular, in a code with minimum pairwise distance between codewords $d$, the update-efficiency has to be at least $d$, because the nearest codeword is at least distance $d$ away. That is, if the update-efficiency of the code $\mathcal{C}$ is denoted by $t(\mathcal{C})$, then

$$t(\mathcal{C}) \geq d(\mathcal{C}),$$

where $d(\mathcal{C})$ is the minimum distance of the code. The purpose of this section is to establish that the above bound can be achieved with the best possible parameters of a linear code. We have seen in Section [II] that for a linear code $\mathcal{C}$, the update-efficiency is simply the weight of the maximum weight row of a generator matrix. The following theorem is from [9].

**Theorem 13:** Any binary linear code of length $n$, dimension $k$ and distance $d$ has a generator matrix consisting of rows of weight $\leq d + s$, where

$$s = \left( n - \sum_{j=0}^{k-1} \left\lceil \frac{d}{2^j} \right\rceil \right)$$

is a nonnegative integer.

The fact that $s$ is a non-negative integer also follows from the well-known Griesmer bound [23], which states that for any linear code with dimension $k$, distance $d$, and length $n \geq \sum_{j=0}^{k-1} \left\lceil d/2^j \right\rceil$.

**Corollary 14:** For any linear $[n, k, d]$ code $\mathcal{C}$ with update-efficiency $t$,

$$d \leq t \leq d + \left( n - \sum_{j=0}^{k-1} \left\lceil \frac{d}{2^j} \right\rceil \right).$$

It is clear that for codes achieving the Griesmer bound with equality, the update-efficiency is exactly equal to the minimum distance, i.e., the best possible. There are a number of families of codes that achieve the Griesmer bound. For examples of such families and their characterizations, we refer the reader to [4], [13].

**Example:** Suppose $\mathcal{C}$ is a $[n = 2^m - 1, k = 2^m - 1 - m, 3]$ Hamming code. For this code

$$t(\mathcal{C}) \leq 3 + (n - 3 - 2 - (k - 2)) = n - k = m = \log(n + 1).$$
One can easily achieve update-complexity $1 + \log(n + 1)$ for Hamming codes. Simply bring any $k \times n$ generator matrix of Hamming code into systematic form, resulting in the maximum weight of a row being bounded above by $1 + (n - k) = 1 + \log(n + 1)$. This special case was mentioned in [2]. This can also be argued from the point of view that as the generator polynomial of a Hamming code (cyclic code) has degree $m$, the maximum row-weight of a generator of a Hamming code will be at most $m + 1 = \log(n + 1) + 1$.

However, we can do even better by explicitly constructing a generator matrix for the Hamming code in the following way. Let us index the columns of the generator matrix by $1, 2, \ldots, 2^m - 1$, and use the notation $(i, j, k)$ to denote the vector with exactly three 1’s, located at positions $i, j,$ and $k$. Then, the Hamming code has a generator matrix given by the row vectors $(i, 2^j, i + 2^j)$ for $1 \leq j \leq m - 1, 1 \leq i < 2^j$. This shows that for all $n$, Hamming codes have update-efficiency only 3. To prove this without explicitly constructing a generator matrix, and to derive some other consequences, we need the following theorem by Simonis [29].

**Theorem 15:** Any $[n, k, d]$ binary linear code can be transformed into a code with the same parameters that has a generator matrix consisting of only weight $d$ rows.

The implication of this theorem is the following: if there exists an $[n, k, d]$ linear code, then there exists an $[n, k, d]$ linear code with update-efficiency $d$. The proof of [29] can be presented as an algorithm that transforms any linear code, given its parameters $[n, k, d]$ and a generator matrix, into an update-efficient linear code (a code with update-efficiency equal to the minimum distance). The algorithm, in time possibly exponential in $n$, produces a new generator matrix with all rows having weight $d$. It is of interest to find a polynomial time (approximation) algorithm for the procedure. That is a generator matrix with all rows having weight within $d(1 + \epsilon)$ for some small $\epsilon$.

On the other hand, the above theorem says that there exists a linear $[n = 2^m - 1, k + 2^m - 1 - m, 3]$ code that has update-efficiency only 3. All codes with these parameters are equivalent to the Hamming code of same parameters up to a permutation of coordinates [15], providing an alternate proof that Hamming codes have update-efficiency 3.

Analysis of update-efficiency for BCH codes and other linear codes is of independent interest.
In general, finding a sparse basis for a linear code given its generator matrix seems to be a hard problem, although the actual complexity class of the problem merits further investigation. Recently, a sparse basis is presented for 2-error-correcting BCH codes in [12].

B. Correcting adversarial errors with a randomized code

Although it is impossible for a fixed error-correction code with small update-efficiency to correct a large number of errors, if we randomize the code, then it is possible to fool the adversary. In fact, with a randomized code it is possible to correct \( p n \) adversarial errors with a code rate close to the capacity of BSC(\( p \)). This idea has been used in the case of erasures in [2]. Let \((\hat{C}, \hat{\phi})\) be a random code derived as follows from another code \((C, \phi)\). Suppose \( \sigma \in S_n \) is an uniform random permutation on the set \( \{1, \ldots, n\} \), and \( z \in \mathbb{F}_2^n \) is a uniform random vector. The random encoding in \( \hat{C} \) is defined by \( \hat{\phi}(x) = \sigma(\phi(x)) + z, x \in \mathbb{F}_2^k \). If the operation of the decoding algorithm of \( C \) and \( \hat{C} \) are denoted by \( \psi \) and \( \hat{\psi} \) respectively, then \( \hat{\psi}(y) = \psi(\sigma^{-1}(y + z)), y \in \mathbb{F}_2^n \).

We have the following theorem that stems from [21].

**Theorem 16:** Let \( C \) be a code with rate \( 1 - h_B(p) - \epsilon \) that achieves probability of error 0 as \( n \to \infty \) over BSC(\( p \)). Suppose \( \hat{C} \) is a random code formed as above. Then, against any adversarially chosen \( p n \) errors, the code \( \hat{C} \) will have probability of error approaching 0 as \( n \to \infty \). In the above theorem, if we take the code \( C \) to be the code designed in Section [III], then the code \( \hat{C} \) remains an update-efficient code with update-efficiency \( \log n \). Hence, by sharing \( O(n \log n) \) bits between the encoder and decoder, it is possible to correct a large number of adversarial errors with a high rate code. We omit the proof of the above theorem here, as it follows directly from [21].

C. General update-efficient codes

Let us now give a more general definition of update-efficiency that we started with in the introduction.

**Definition 3:** A code is called \((u, t)\)-update-efficient if, for any \( u \) bit changes in the message, the codeword changes by at most \( t \) bits. In other words, the code \((C, \phi)\) is \((u, t)\)-update-efficient
if for all $x \in \mathbb{F}_2^k$, and for all $e \in \mathbb{F}_2^k : \text{wt}(e) \leq u$, we have $\phi(x + e) = \phi(x) + e'$, for some $e' \in \mathbb{F}_2^n : \text{wt}(e') \leq t$.

It is easy to see that an $(1, t)$-update-efficient code is a code with update-efficiency $t$. As discussed earlier, any $(u, t)$-update-efficient code must satisfy $t > d$, the minimum distance of the code. In fact, we can make a stronger statement.

**Proposition 17:** Suppose an $(u, t)$-update-efficient code of length $n$, dimension $k$, and minimum distance $d$ exists. Then,

$$\sum_{i=0}^{u} \binom{k}{i} \leq B(n, d, t),$$

where $B(n, d, w)$ is the size of the largest code with distance $d$ such that each codeword has weight at most $w$.

**Proof:** Suppose $C$ is an update-efficient code, where $x \in \mathbb{F}_2^k$ is mapped to $y \in \mathbb{F}_2^n$. The $\sum_{i=0}^{u} \binom{k}{i}$ different message vectors within distance $u$ from $x$ should map to codewords within distance $t$ from $y$. Suppose these codewords are $y_1, y_2, \ldots$. Consider the vectors $y - y, y_1 - y, y_2 - y, \ldots$. These must be at least distance $d$ apart from one another and all of their weights are at most $t$. This proves the claim.

There are a number of useful upper bounds on the maximum size of constant weight codes (i.e., when the codewords have a constant weight $t$) that can be used to upper bound $B(n, d, t)$. Perhaps the most well-known bound is the Johnson bound [16]. An easy extension of this bound says $B(n, d, t) \leq dn/(dn - 2tn + 2t^2)$, as long as the denominator is positive. However, this bound is not very interesting in our case, where we have $n \gg t \geq d$. The implications of some other bounds on $B(n, d, t)$ on the parameters of update-efficiency is a topic of independent interest.

Note that any code with update-efficiency $t$ is a $(u, ut)$-update-efficient code. Hence, from Section III, we can construct an $(u, O(u \log n))$ update-efficient code that achieves the capacity of a BSC($p$). On the other hand one expects an converse result of the form

$$\sum_{i=0}^{u} \binom{k}{i} \leq K(n, t, p),$$
where $K(n, t, p)$ is the maximum size of a code with codewords having weight bounded by $t$ that achieves arbitrarily small probability of error. Indeed, just by emulating the proof of Theorem 5 we obtain the following result.

**Theorem 18:** Consider using some (possibly non-linear) $(u, t)$-update-efficient code of length $n$, and dimension (possibly fractional) $k$ over BSC($p$). Assume that

$$t \leq \frac{(1 - \alpha) \log \sum_{i=0}^{u} \binom{k}{i}}{\log(1 - p)/p},$$

for any $\alpha > 0$. Then, the average probability of error is at least $1 - o(1)$, where $o(1)$ denotes a quantity that goes to zero as $k \to \infty$.

This shows that the $(u, O(u \log n))$ update-efficient code constructed by the method of Section III is almost optimal for $u \ll n$.

**Remark 4 (Bit error rate and error reduction codes):** Suppose we change the model of update-efficient code in the following way (limited to only this remark). The encoding $\phi : \mathbb{F}_2^k \to \mathbb{F}_2^n$ and decoding $\theta : \mathbb{F}_2^n \to \mathbb{F}_2^k$, is such that for a random error vector $e$ induced by the BSC($p$) and any $x \in \mathbb{F}_2^n$, $d_H(\theta(\phi(x) + e), x) \sim o(k)$ with high probability. This can be thought of as an error-reducing code or a code with low message bit error rate [20]. Under this notion, error-reducing codes are update-efficient. When the message changes $\leq u$ bits from the previous state $x \in \mathbb{F}_2^k$, we do not change the codeword. Then, the decoder output will be within $o(k) + u$ bits from the original message.

**VII. Rate-Distortion Counterparts and Concluding Remarks**

In this paper, we have focused on error correcting codes possessing good update efficiency and local recovery properties. In principle, these properties are also applicable to the problem of lossy source coding. Update-efficient codes for lossless source compression have been considered before in several papers, e.g., [24].

The standard formulation of lossy source coding in terms of a source and distortion model can be found in any standard textbook on information theory (e.g., [8]). The associated rate-distortion function $R(D)$ expresses the optimal (smallest) rate achievable given a normalized distortion $D$. 
Update-efficiency and local recoverability have natural analogs for lossy source codes. In more detail, update-efficiency can be measured by asking how much the encoding (compression) of the source changes when the source is changed slightly, e.g., how many bits of the compression change when 1 bit of the original source is changed. Local recoverability for a lossy source code can be measured by the number of bits of the compression that must be queried in order to recover any particular symbol of reconstruction. That is, a lossy source code has good local recoverability if, for all indices \( i \), few bits of the compression must be read in order to compute the \( i^{th} \) symbol of the lossy reconstruction.

The main questions, in the spirit of this paper, to be asked are, if we allow a rate slightly above the rate-distortion function, i.e., rate \( R(D) + \epsilon \), what is the best possible local recoverability, and what is the best possible update-efficiency (in terms of \( \epsilon \))? As a simple example, we briefly consider these questions in the context of compressing a uniform binary source under Hamming distortion. In this case, it can be shown that local recoverability must grow as \( \Omega(\log(1/\epsilon)) \). This is a corollary of results for LDGM codes (Theorem 5.4.1 from [7]), and the proof given there already applies to non-linear codes. LDGM codes show that \( O(\log(1/\epsilon)) \) recovery complexity is achievable. Thus, in this simple case, the local recoverability can be characterized up to a constant factor.

Update-efficiency, on the other hand, remains an open question, even for this simple model. Update-efficiency of \( O(1/\epsilon \log(1/\epsilon)) \) can be achieved via random codes, but it is unclear that this is optimal. In particular, it is unclear that the update-efficiency has to scale with \( \epsilon \) at all. We note that for a more general rate-distortion problem, random coding would only achieve update-efficiency \( O(1/\epsilon^2) \), i.e., for a general discrete memoryless source and additive distortion model, random codes would require a block length of \( O(1/\epsilon^2) \) to come within \( \epsilon \) of the rate-distortion function. However, for the special case of a uniform binary source under Hamming distortion, it is easily verified that a random code of length \( O(1/\epsilon \log(1/\epsilon)) \) comes within \( \epsilon \) of the rate-distortion function. Therefore, we can split an \( n \)-bit source into blocks of length \( O(1/\epsilon \log(1/\epsilon)) \), and apply the random code constructed above to each block. Clearly, changing one bit of the source can change at most \( O(1/\epsilon \log(1/\epsilon)) \) bits of the compression, and the code
achieves the same distortion and compression rate as the base random code. This shows that $O(1/\epsilon \log(1/\epsilon))$ update-efficiency can be achieved for this particular model. A more detailed study of local recoverability and update-efficiency for lossy source coding is left for future work.

ACKNOWLEDGMENT

A. M. thanks Alexander Barg, Yury Polyanskiy and Barna Saha for useful discussions.

REFERENCES

[1] N. Alon, J. Spencer, The Probabilistic Method, Wiley-Interscience, 2000.
[2] N. P. Anthapadmanabhan, E. Soljanin, S. Vishwanath, “Update-efficient codes for erasure correction,” 48th Annual Allerton Conference on Communication, Control, and Computing, pp. 376–382, October, 2010.
[3] A. Barg, G. D. Forney, Jr., “Random codes: minimum distances and error exponents,” IEEE Transactions on Information Theory, vol. 48, no. 9, pp. 2568–2573, September, 2002.
[4] B. I. Belov, V. N. Logachev, V. P. Sandimirov, “Construction of a class of linear binary codes achieving the Varshamov/Griesmer bound,” Problema Peredachi Informatsii, vol. 10, issue 3, pp. 36–44, 1974.
[5] V. Cadambe, A. Mazumdar, “On locally repairable codes over arbitrarily sized alphabet,” International Symposium on Network Coding (NetCod), Calgary, Canada, 2013.
[6] N. Calkin, “Dependent sets of constant weight binary vectors,” Combinatorics, Probability and Computing, vol. 6, no. 3, pp. 263–271, 1997.
[7] V. Chandar, Sparse Graph Codes for Compression, Sensing, and Secrecy, Ph.D. Thesis, MIT, 2010.
[8] T. Cover, J. Thomas, Elements of Information Theory, 2nd Ed., Wiley-Interscience, 2006.
[9] S. D. Dodunekov, N. L. Manev, “An improvement of the Griesmer bound for some small minimum distances,” Discrete Applied Mathematics, vol. 12, pp. 103–114, 1985.
[10] R. G. Gallager, Low Density Parity Check Codes, Monograph, M.I.T. Press, 1963.
[11] P. Gopalan, C. Huang, H. Simitci, S. Yekhanin, “On the locality of codeword symbols,” IEEE Transactions on Information Theory, vol. 58, pp. 6925–6934, Nov 2012.
[12] E. Grigorescu, T. Kaufman, “Explicit low-weight bases for BCH codes,” IEEE Transactions on Information Theory, vol. 58, no. 1, pp. 78–81, Jan 2012.
[13] T. Helleoeth, “New constructions of codes meeting the Griesmer bound,” IEEE Transactions on Information Theory, vol. 29, no. 3, May, 1983.
[14] E. Hof, I. Sason, S. Shamai, “Performance bounds for nonbinary linear block codes over memoryless symmetric channels,” IEEE Transactions on Information Theory, vol. 55, no. 3, pp. 977–996, 2009.
[15] W. C. Huffman, V. Pless, Fundamentals of Error-Correcting Codes, Cambridge, 2003.
[16] S. M. Johnson, “A new upper bound for error-correcting codes,” *IRE Transactions on Information Theory*, pp. 203–207, April, 1962.

[17] A. Jule, I. Andriyanova, “Some results on update complexity of a linear code ensemble,” *International Sym. on Network Coding*, pp. 1–5, 2011.

[18] A. M. Kakhaki, H. K. Abadi, P. Pad, H. Saeedi, F. Marvasti, K. Alishahi, “Capacity achieving linear codes with binary sparse generating matrices,” preprint, arXiv:1102.4099, 2011.

[19] G. M. Kamath, N. Prakash, V. Lalitha, P. V. Kumar, N. Silberstein, A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath, “Explicit mbr all-symbol locality codes,” arXiv preprint, arXiv:1302.0744, 2013.

[20] Y. Kochman, A. Mazumdar, Y. Polyanskiy, “Results on combinatorial joint source-channel coding,” *Information Theory Workshop*, Lausanne, Switzerland, 2012.

[21] M. Langberg, “Private codes or Succinct random codes that are (almost) perfect,” Proc. Foundations of Computer Science, pp. 325–334, 2004.

[22] S. Litsyn, V. Shevelev, “On ensembles of low-density parity-check codes: asymptotic distance distributions,” *IEEE Transactions on Information Theory*, vol. 48, no. 4, pp. 887–908, April 2002.

[23] F. Macwilliams, N. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1977.

[24] A. Montanari, E. Mossel, “Smooth compression, Gallager bound and nonlinear sparse-graph codes,” in *Proc. 2008 IEEE Intl. Symp. on Information Theory (ISIT ’08)*, Piscataway, NJ: IEEE Press, 2008, pp. 2474-2478.

[25] D. Papailiopoulos, A. G. Dimakis, “Locally repairable codes,” *IEEE International Symposium on Information Theory (ISIT)*, 2012, pp. 2771–2775.

[26] A. S. Rawat, S. Vishwanath, A. Bhowmick, E. Soljanin, “Update-efficient codes for distributed storage,” *IEEE International Symposium on Information Theory*, pp. 1457–1461, Jul 31–Aug 5, 2011.

[27] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath, “Optimal locally repairable and secure codes for distributed storage systems,” arXiv preprint, arXiv:1210.6954, 2012.

[28] R. M. Roth, *Introduction to Coding Theory*, 1st Ed., Cambridge, 2006.

[29] J. Simonis, “On generator matrices of codes,” *IEEE Transactions on Information Theory*, vol. 38, no. 2, March, 1992.

[30] A. Smith, “Scrambling adversarial errors using few random bits, optimal information reconciliation, and better private codes,” Symposium on Discrete Algorithms (SODA), pp. 395–404, 2007.

[31] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, “Optimal locally repairable codes and connections to matroid theory,” arXiv preprint, arXiv:1301.7693, 2013.