The Minimum Distance of Turbo-Like Codes

Louay Bazzi *, Mohammad Mahdian †, Daniel A. Spielman ‡

May 6, 2003

Abstract

We derive worst-case upper bounds on the minimum distance of parallel concatenated Turbo codes, serially concatenated Turbo codes, repeat-accumulate codes, repeat-convolute codes, and generalizations of these codes obtained by allowing nonlinear and large-memory constituent codes. We show that parallel-concatenated Turbo codes and repeat-convolute codes with sub-linear memory are asymptotically bad. We also show that depth-two serially concatenated codes with constant-memory outer codes and sub-linear-memory inner codes are asymptotically bad. In contrast, we prove that depth-three serially concatenated codes obtained by concatenating a repetition code with two accumulator codes through random permutations can be asymptotically good. We also show how to construct interleavers for Turbo codes with two-branches that maximize their minimum distance up to a constant factor.

1 Introduction

The low-complexity and near-capacity performance of Turbo codes [3, 10] has led to a revolution in coding theory. The most famous casualty of the revolution has been the idea that good codes should have high minimum distance: the most useful Turbo codes have been observed to have low minimum distance. In this work, we provide general conditions under which many constructions of turbo-like codes, including families of

*Department of Electrical Engineering and Computer Science, MIT, Cambridge, Massachusetts 02139. Supported by ARO grant DAA L03-92-G-0115, and MURI grant DAAD19-00-1-0466.

†Department of Mathematics, MIT, Cambridge, Massachusetts 02139. Work on this paper supported by NSF CCR: 9701304.

‡Department of Mathematics, MIT, Cambridge, Massachusetts 02139. Work on this paper supported by NSF CCR: 9701304.

serially-concatenated Turbo-like codes [2] and Repeat-Accumulate (RA) codes [5, 7, 8], must be asymptotically bad. We also present a simple family of depth-3 serially concatenated Turbo-like codes that are asymptotically good.

Our work is motivated by the analyses of randomly constructed parallel and serially concatenated Turbo codes by Kahale and Urbanke [8] and of Turbo codes with two branches by Breiling [4]. Kahale and Urbanke provided estimates on the probable minimum distance of randomly generated parallel concatenated Turbo codes with a constant number of branches. They also provided similar estimates for the minimum distance of the random concatenation of two convolutional codes with bounded memory. Breiling proved that the parallel concatenation of two convolutional codes with bounded memory always has logarithmic minimum distance. We note that both of these bounds are for linear codes with low memory.

These analyses naturally lead to the following five questions:

(better than random?) Do there exist asymptotically good parallel concatenated Turbo codes with more than two branches or do there exist asymptotically good repeat-convolute or repeat-accumulate codes?

Note that the result of Breiling only applies to Turbo codes with two branches and the results of Kahale and Urbanke do not preclude the existence of codes that are better than the randomly generated codes.

(larger memory?) What happens if we allow the memories of the constituent convolutional codes to grow with the block length?

All the previous bounds become vacuous if the memory even grows logarithmically with the block length.

(non-linearity?) Can the minimum distance of Turbo-like codes be improved by the use of non-linear constituent encoders, such as automata encoders?

(concatenation depth?) Can one obtain asymptotically good codes by serially concatenating a repetition code with two levels of convolutional codes?

(explicit interleavers?) Can one construct an interleaver under which Turbo codes with two branches have logarithmic minimum distance?

We will give essentially negative answers to the first three questions and positive answers to the last two. For parallel concatenations and depth-2 serial concatenations of convolutional and automata codes, we prove upper bounds on the minimum distance of the resulting codes in terms of the memories of the constituent codes. We show that parallel concatenated codes and repeat-convolute codes are asymptotically bad if their

constituent codes have sub-linear memory. This bound even holds if the constituent codes are non-linear. If we assume that the constituent codes are linear, then we obtain upper bounds that almost match the high-probability upper bounds for random codes obtained by Kahale and Urbanke. We also show that depth-two serially concatenated convolutional codes are asymptotically bad if their inner code has sub-linear memory and their outer code has constant memory. In contrast, we show that depth-three concatenations of constant-memory codes can be asymptotically good. In particular, we prove this for the random concatenation of a repetition code with two accumulator codes.

1.1 Turbo-like codes

The fundamental components of the codes we consider in this paper are convolutional codes and their non-linear generalizations, which we call automata codes. The fundamental parameter of a convolutional code that we will measure is its *memory*—the number of registers in its encoder. The memory can also be defined to be the binary logarithm of the number of states in the encoder's state diagram. A general automata encoder is obtained by considering an encoder with any deterministic state diagram. We will consider automata encoders that read one bit at each time step, and output a constant number of bits at each time step. These are also described as deterministic automata or transducers with one input bit and a constant number of output bits on each transition. We will again define the memory of an automata encoder to be the binary logarithm of its number of states.

Given a k convolutional codes $Q_1, \ldots, Q_k$, a message length n, and k permutations $\pi_1, \ldots, \pi_k$ of length n, we can define the *parallel concatenated Turbo code with k branches* [3, 10] $P_{Q_1,\ldots,Q_k,\pi_1,\ldots,\pi_k}$, to be the code that encodes a binary message x to $(x, Q_1(\pi_1(x)), \ldots, Q_k(\pi_k(x)))$, where $\pi_i(x)$ denotes the permutation of the bits in x according to π_i and $Q_i(y)$ denotes the output of the convolutional code Q_i on input y.

Given an integer k, we define the repeat-k-times code, r_k, to be the code that just repeats each of its input bits k times. Given a convolutional code Q, a message length n, and a permutation π of length kn, we define the *repeat-convolute code* [5], $C_{k,\pi,Q}$ to be the code that maps an input $x \in \{0,1\}^n$ to $(x, Q(\pi(r_k(x))))$. That is, each bit of the input is repeated k times, the resulting kn bits are permuted, and then fed through the convolutional encoder. We also assume that the input x is output as well. While some implementations do not include x in the output, its exclusion cannot improve the minimum distance so we assume it appears. The number k is called the *repetition factor* of the code. When the convolutional code Q is the accumulator (*i.e.,* the map $Q(x)_j = \sum_{i=1}^{j} x_i$), this code is called a *repeat-accumulate (RA) code* [5]. We remark that a parallel concatenated Turbo code with k branches can be simulated by a repeat-

convolute code with repetition factor k whose encoder is a product of the encoders in the parallel code.

Given two convolutional encoders Q_o and Q_i that output h_o and h_i bits per time step respectively, an integer n, and a permutation π of length $h_o n$, we define the depth-two *serial concatenated Turbo code* [2, 10] $C_{Q_o,Q_i,\pi}$ to be the rate $1/h_o h_i$ code that maps an input $x \in \{0,1\}^n$ to the codeword $Q_i(\pi(Q_o(x)))$. The codes Q_o and Q_i are called *outer* and *inner* codes, respectively. A classical example of serially concatenated Turbo codes, and that considered in [8], is a rate 1/4 code given by the map $(\pi(x, L_o(x)), L_i(\pi((x, L_o(x)))))$, where L_o and L_i are rate-1 convolutional codes. This fits into our framework with $Q_o(x) = (x, L_o(x))$ and $Q_i(x) = (x, L_i(x))$.

One can allow greater depth in serial concatenation. The only codes of greater depth that we consider will be repeat-accumulate-accumulate codes (RAA). These are specified by a repetition factor k, an integer n, and two permutations π_1 and π_2 of length kn. Setting Q_1 and Q_2 to be accumulators, the resulting code maps an input x to $Q_2(\pi_2(Q_1(\pi_1(r_k(x)))))$.

We can generalize each of these constructions by allowing the component codes to be automata codes. In this case, we will refer to the resulting codes as *parallel concatenated Turbo-like codes*, *repeat convolute-like codes*, and *serially concatenated Turbo-like codes*. We refer to all the codes in this family as *Turbo-like* codes.

In practice, some extra bits are often appended to the input x of a Turbo-like code so as to guarantee that some of the encoders return to the zero state. As this addition does not substantially increase the minimum distance of the resulting code, we will not consider this technicality in this paper.

1.2 Previous results

Kahale and Urbanke [8] proved that if one builds a parallel concatenated Turbo code from a random interleaver and convolutional encoders of memory at most M, then the resulting code has minimum distance at most $\tilde{O}(2^M n^{1-2/k})$ [1] and at least $\Omega(n^{1-2/k})$ with high probability For rate 1/4 serially concatenated Turbo codes of the form mentioned in the previous section with a random interleaver, they proved that the resulting code has minimum distance at most $\tilde{O}(2^{M_i} n^{1-2/d_o})$ and at least $\Omega(n^{1-2/d_o})$ with high probability, where d_o is the free distance of the outer code and M_i is the inner code memory.

For parallel concatenated Turbo codes with two branches, Breiling [4] proved that no construction could be much better than a random code: if the constituent codes have memory M, then the minimum distance of the resulting code is $O(2^M \log n)$.

[1] $\tilde{O}(f(n))$ means $O(f(n)\log^{O(1)} n)$.

Serially concatenated codes of depth greater than 2 were studied by Pfister and Siegel [9], who performed experimental analyses of the serial concatenation of repetition codes with l levels of accumulators connected by random interleavers, and theoretical analyses of of concatenations of a repetition code with certain rate-1 codes for large l. Their experimental results indicate that the average minimum distance of the ensemble starts becoming good for $l \geq 2$, which is consistent with our theorem. For certain rate-1 codes and l going to infinity, they proved their codes could become asymptotically good.

In many arguments, we use techniques introduced Ajtai [1] to prove time-space trade-offs for branching programs.

1.3 Our results

In Section 2.1, we upper bound the minimum distance of repeat-convolute-like codes. We prove that repeat-convolute-like codes of message length n, memory M, and repetition factor k have minimum distance at most $O(n^{1-1/k}M^{1/k})$, and therefore such codes are asymptotically bad when k is constant and M is sub-linear in n. Note that M sub-linear in n corresponds to the case when the size of the corresponding trellis is sub-exponential, and so it includes the cases in which the codes have natural sub-exponential time iterative decoding algorithm. As parallel concatenated Turbo-like codes in which the component codes have memory M can be encoded by repeat-convolute codes with memory kM, we find that these are also asymptotically bad for k constant and M sub-linear in n. This proof uses techniques introduced by Ajtai [1] for obtaining time-space trade-offs for branching programs. Comparing our upper bound with the $\tilde{O}(2^M n^{1-2/k})$ high-probability upper bound of Kahale and Urbanke for parallel concatenated codes, we see that our bound has a much better dependence on M and a slightly worse dependence on k. A similar relation holds between our bound and the $O(2^M \log n)$ upper bound of Breiling [4].

In Section 2.1, we restrict our attention to linear repeat-convolute codes, and obtain upper bounds on the minimum distance of $O(2^{2M} n^{1-1/\lceil k/2 \rceil} \log n)$ for codes with repetition factor k and memory M. For even k, this bound is very close to the high-probability bound of Kahale and Urbanke.

In Section 2.3, we show how to construct an interleaver for parallel concatenated Turbo codes with two branches and accumulator constituent codes for which the resulting code has minimum distance at least $\frac{1}{2} \log n$ for sufficiently large n. This matches the upper bound of Breiling up to a constant. Our construction is motivated by a technique introduced by Gallager [6] to construct low-density parity-check codes.

In Section 3.1, we study serially concatenated Turbo-like codes with two levels, and prove

that if the outer code has memory M_o and the inner code has memory M_i, then the resulting code has minimum distance at most $O(n^{1-1/h_o(M_o+2)} M_i^{1/h_o(M_o+2)})$. Accordingly, we see that such codes are asymptotically bad when M_o, h_o and h_i are constants and M_i is sub-linear in n. The proof uses similar techniques to those used in Section 2.1. When specialized to the classical rate 1/4 construction of serially concatenated Turbo codes considered by Kahale and Urbanke [8], our bounds on the minimum distance becomes $O(n^{1-1/(2M_o+4)} M_i^{1/(2M_o+4)})$. Comparing this with the high-probability $\tilde{O}(n^{1-2/d_o} 2^{M_i})$ upper bound of Kahale and Urbanke, we see that our bound is better in terms of M_i, comparable in terms of d_o, and close to the existential bound.

Finally, in Section 3.2, we show that serially concatenated codes of depth greater than two can be asymptotically good, even if the constituent codes are repetition codes and accumulators. In particular, we prove that randomly constructed RAA codes are asymptotically good with constant probability.

We conclude with some open questions.

2 Repeat-convolute-like codes

In this section we consider codes that are obtained by serially concatenating a repeat-k-times code r_k with any code Q that can be encoded by an automata (transducer) with at most 2^M states and one output bit per transition. More precisely, if Q is such an encoder, π is a permutation, and r_k is the repeat-k-times map, we define $C_{k,\pi,Q}$ to be the code that maps a string x to $C_{k,\pi,Q}(x) := (x, Q(\pi(r_k(x))))$.

This class of codes contains repeat-convolute codes and repeat-accumulate code when Q is a convolutional code. It also contains parallel concatenated Turbo codes: a parallel concatenated Turbo code with k branches and memory M can be encoded by a repeat-convolute-like code with repetition factor k and memory kM by interleaving the permutations on the k branches.

2.1 An upper bound on the minimum distance

Theorem 1 *Let $k \geq 2$ be a constant integer, Q an automata encoder with at most 2^M states, n an integer, and π a permutation of length kn.*

If $n \geq 2^k kM$, then the minimum distance of the code $C_{k,\pi,Q}$ is at most

$$3k^2 n^{1-1/k} M^{1/k} + 2^k kM + k + 1.$$

Proof: To prove this theorem, we make use of the techniques introduced by Ajtai [1] for proving time-space trade-offs for branching programs. In particular, for an input x of length n, the encoding action of Q is naturally divided into kn time steps in which the automata reads a bit of $\pi(x)$, outputs a bit, and changes state. For convenience, we will let $I = \{1, \ldots, kn\}$ denote the set of time steps, and we will let $s_i(x)$ denote the state of Q on input $\pi(r_k(x))$ at the end of the i'th time step.

Let C denote the code $C_{k,\pi,Q}$. To prove the claimed bound on the minimum distance of C, we will prove the existence of two input strings, x and y, a set $U \subset \{1, \ldots, n\}$ of size at most $2^k k(M+1)$, and $J \subset I$ of size at most $3k^2 n^{1-1/k}(M+1)^{1/k} + k$ such that x and y may only differ on bits with indices in U and $s_i(x)$ and $s_j(x)$ may only differ on time steps with indices in J.

To construct the set J, we first divide the set of time steps I into b consecutive intervals, where b is a parameter we will specify later. We choose these intervals so that each has size $\lfloor kn/b \rfloor$ or $\lceil kn/b \rceil$. For example, if $k = 2$, $n = 4$, and $b = 3$ we can divide $I = \{1, \ldots, 8\}$ into the intervals $[1, 3]$, $[4, 6]$, and $[7, 8]$.

For each index of an input bit $i \in \{1, \ldots, n\}$, we let S_i denote the multiset of time intervals in which Q reads input bit i (this is a multiset as a bit can appear multiple times in same interval). As each bit appears k times, the multisets S_i each have size k. As there are b intervals, there are at most b^k possible k-multisets of intervals. So, there exists a set $U \subset \{1, \ldots, n\}$ of size at least n/b^k and a multiset of intervals, S, such that for all $i \in U$, $S_i = S$. Let U be such a set with $|U| = \lceil n/b^k \rceil$ and let T be the corresponding set of intervals. Let $l = |T|$. The set J will be the union of the intervals in T.

Let $t_1, \ldots, t_l$ be the last times in the time intervals in T (e.g., in the above example the last time of the interval $[4, 6]$ is 6). For each $x \in \{0,1\}^n$, that is zero outside U, we consider the vector of states of Q at times $t_1, \ldots, t_l$ on input $\pi(r_k(x))$: $\{s_{t_i}(x)\}_{i=1}^{l}$. As the number of such possible sequences is at most 2^{Ml} and the number of x that are zero outside U is $2^{|U|}$, if

$$2^{|U|} > 2^{Ml}, \tag{1}$$

then there should exist two different strings x and y that are both zero outside of U and such that $s_{t_i}(x) = s_{t_i}(y)$ for $i = 1, \ldots, l$. To make sure that (1) is satisfied, we set

$$b = \left\lceil \left(\frac{n}{kM} \right)^{1/k} \right\rceil - 1.$$

Our assumption that $n \geq 2^k kM$ ensures that $b \geq 1$. Now, since

- x and y agree outside U,

- the bits in U only appear in time intervals in T, and
- Q traverses the same states at the ends of time intervals in T on inputs $\pi(r_k(x))$ and $\pi(r_k(y))$,

Q must traverse the same states at all times in intervals outside T on inputs $\pi(r_k(x))$ and $\pi(r_k(y))$. Thus, the bits output by Q in time steps outside intervals in T must be the same on inputs $\pi(r_k(x))$ and $\pi(r_k(y))$. So $Q(\pi(r_k(x)))$ and $Q(\pi(r_k(y)))$ can only disagree on bits output during times in the intervals in T, and hence on at most $l\lceil kn/b\rceil$ bits. This means that the distance between $C(x)$ and $C(y)$ is at most

$$\begin{aligned}
|U| + l\lceil kn/b\rceil &\leq \left\lceil \frac{n}{b^k} \right\rceil + k\lceil kn/b\rceil, \text{ as } |U| = \lceil n/b^k\rceil \text{ and } l \leq k, \\
&\leq \frac{n}{b^k} + 1 + \frac{k^2 n}{b} + k \\
&\leq \frac{n}{\left\lceil \left(\frac{n}{kM}\right)^{1/k} - 1 \right\rceil^k} + \frac{k^2 n}{\left(\frac{n}{kM}\right)^{1/k} - 1} + k + 1 \\
&\leq \frac{n}{\left(\left(\frac{n}{kM}\right)^{1/k} - 1\right)^k} + \frac{k^2 n}{\left(\frac{n}{kM}\right)^{1/k}} \frac{\left(\frac{n}{kM}\right)^{1/k}}{\left(\frac{n}{kM}\right)^{1/k} - 1} + k + 1 \\
&\leq \frac{n}{\left(\frac{n}{kM}\right)} \left(\frac{\left(\frac{n}{kM}\right)^{1/k}}{\left(\frac{n}{kM}\right)^{1/k} - 1} \right)^k + \frac{k^2 n}{\left(\frac{n}{kM}\right)^{1/k}} \frac{\left(\frac{n}{kM}\right)^{1/k}}{\left(\frac{n}{kM}\right)^{1/k} - 1} + k + 1 \\
&\leq \frac{n}{\left(\frac{n}{kM}\right)} 2^k + 2\frac{k^2 n}{\left(\frac{n}{kM}\right)^{1/k}} + k + 1, \text{ as } n \geq 2^k kM \\
&\leq 2^k kM + 2k^2 n^{1-1/k} M^{1/k} k^{1/k} + k + 1, \\
&\leq 3k^2 n^{1-1/k} M^{1/k} + 2^k kM + k + 1,
\end{aligned}$$

as $k^{1/k} \leq 3/2$. ■

Corollary 1 *Let k be a constant. Then, every repeat-convolute code with input length n and memory M and repetition factor k and every parallel concatenated Turbo code with input length n, convolutional encoder memory M and k branches has minimum distance $O(n^{1-1/k}M^{1/k})$. Thus, such codes cannot be asymptotically good for M sub-linear in n.*

This means that if we allow M to grow like $\log n$, or even like $n^{1-\epsilon}$ for some $\epsilon > 0$, the minimum relative distance of the code will still go to zero. Moreover, M sub-linear in n

corresponds to the case in which the size of the corresponding trellis is sub-exponential, and therefore it includes all the cases in which such codes have sub-exponential-time iterative decoding algorithms.

It is interesting to compare our bound with that obtained by Kahale and Urbanke [8], who proved that a randomly chosen Turbo code with k branches has minimum distance $\tilde{O}(2^M n^{1-2/k})$ with high probability. Theorem 1 has a much better dependence on M and a slightly worse dependence on n. A similar comparison can be made with the bound of Breiling [4], who proved that every parallel concatenated Turbo code with $k = 2$ branches has minimum distance $O(2^M \log n)$. In the next section, we prove an upper bound whose dependence on M and n is similar to that obtained by these authors.

2.2 Improving the bound in the linear, low-memory case

We now prove that every repeat-convolution code with repetition factor k and memory M has minimum distance at most $O(n^{1-1/\lceil k/2 \rceil} 2^{2M})$. We note that this bound matches the bound of Breiling for parallel concatenated Turbo codes with two branches [4] (*i.e.* when $k = 2$), and for even k matches the bound proved for randomly constructed parallel-concatenated Turbo codes proved by Kahale and Urbanke [8]. As Kahale and Urbanke proved similar lower bounds for $k \geq 3$, we learn that the minimum distances of randomly constructed Turbo codes is not too different from that of optimally constructed Turbo codes.

Theorem 2 *Let $k \geq 2$ and n be integers, let π be a permutation of length kn, and let Q be a recursive convolutional encoder with memory M. Assuming $M < (\log n - 3)/k$, the minimum distance of the repeat-convolute code $C_{k,\pi,Q}$ is at most*

$$16k^2 n^{1-1/\lceil k/2 \rceil} 2^{2M} \log n + 6 \log n.$$

Thus, when k is constant, the minimum distance of the repeat-convolute code $C_{k,\pi,Q}$ is

$$O(n^{1-1/\lceil k/2 \rceil} 2^{2M}).$$

We note that if the convolutional code is non-recursive, it is trivial to show that on input 10^{n-1} the output codeword will have weight at most $k2^M$.

Our proof of Theorem 2 will make use of the following fact about linear convolutional codes mentioned in Kahale-Urbanke [8]:

Lemma 1 *For any recursive convolutional encoder Q of memory M, there is a number $\delta \leq 2^M$ such that, after processing any input of the form $0^*10^{j\delta-1}1$ for any integer j, Q comes back to the zero state after processing the second 1. In particular, the weight of the output of Q after processing any such input is at most $j\delta$.*

Proof of Theorem 2:

Let δ be the number shown to exist in Lemma 1 for convolutional code Q. As in [8] and [4], we will construct a low-weight input x on which $C_{k,\pi,Q}(x)$ also has low-weight by taking the exclusive-or of a small number of weight 2 inputs each of whose two 1s are separated by a low multiple of δ 0s. As $C_{k,\pi,Q}$ is a linear code, its minimum distance equals the minimum weight of its codewords.

To construct this low-weight input, we first note that every bit of the input x appears exactly k times in the string $\pi(r_k(x))$. For every $i \in \{1,\ldots,n\}$ and every $1 \leq j \leq k$, let $\sigma_j(i)$ denote the position of the j'th appearance of the bit i in $\pi(r_k(x))$. For each bit, i, consider the sequence $(\sigma_1(i) \bmod \delta, \sigma_2(i) \bmod \delta, \ldots, \sigma_k(i) \bmod \delta)$. Since there are at most δ^k such possible sequences and n input bits, there exists a set $U \subset \{1,\ldots,n\}$ of size at least $\lceil n/\delta^k \rceil$ such that all of its elements induce the same sequence. That is, for all i and j in U, $\sigma_l(i) - \sigma_l(j)$ is divisible by δ for all $1 \leq l \leq k$. From now on, we will focus on the input bits with indices in U, and construct a low-weight codeword by setting some of these bits to 1.

As in the proof of Theorem 1, we now partition the set of time steps $\{1,\ldots,kn\}$ into b consecutive intervals, $I_1, I_2, \ldots, I_b$, each of length $\lceil kn/b \rceil$ or $\lfloor kn/b \rfloor$, where b is a parameter we will specify later. For every index $i \in U$, we let the *signature* of i be the k-tuple whose j'th component is the index of the interval to which $\sigma_j(i)$ belongs.

Now, we construct a hypergraph $\mathcal{H}$ as follows: $\mathcal{H}$ has k parts, each part consisting of b vertices which are identified with the intervals $I_1,\ldots,I_b$. There are $|U|$ hyperedges in $\mathcal{H}$, one corresponding to each input bit with index in U. The vertices contained in the hyperedge are determined by the signature of the corresponding bit: if input bit i has signature $(i_1, i_2, \ldots, i_k)$, then the i'th hyperedge contains the i_j'th vertex of the j'th part, for every $j = 1,\ldots,k$. Thus, $\mathcal{H}$ is a k-partite k-uniform hypergraph (i.e., each hyperedge contains exactly k vertices, each from a different part) with b vertices in each part and $|U| \geq n/\delta^k$ edges.

We now define a family of subgraphs such that if $\mathcal{H}$ contains one of these subgraphs, then $C_{k,\pi,Q}$ must have a low-weight codeword. We define an ℓ-forbidden subgraph S of $\mathcal{H}$ to be a set of at least one and at most 2ℓ hyperedges in $\mathcal{H}$ such that each vertex in $\mathcal{H}$ is contained in an even number of the hyperedges of S. (One can think of an ℓ-forbidden subgraph as a generalization of a cycle of length ℓ to hypergraphs). In Lemma 2, we

prove that if $\mathcal{H}$ contains an ℓ-forbidden subgraph then $C_{k,\pi,Q}$ has a codeword of weight at most $2\ell + \ell\lceil kn/b\rceil$. In Lemma 3, we prove that if $\mathcal{H}$ contains at least $4b^{\lceil k/2\rceil}$ edges, then it contains a $k\log b$ forbidden subgraph. As $\mathcal{H}$ has at least n/δ^k edges, if we set

$$b = \left\lfloor \left(\frac{n}{4\delta^k}\right)^{1/\lceil k/2\rceil} \right\rfloor,$$

then $n/\delta^k \geq 4b^{\lceil k/2\rceil}$; so, Lemma 3 will imply that $\mathcal{H}$ has a $k\log b$-forbidden subgraph and Lemma 2 will imply that $C_{k,\pi,Q}$ has a codeword of weight at most $2k\log b + k\log b\lceil kn/b\rceil$. Plugging in the value we have chosen for b, we find that the minimum distance of $C_{k,\pi,Q}$ is at most

$$\begin{aligned}
2k\log b + k\lceil kn/b\rceil \log b &\leq 4\log n + 2\lceil kn/b\rceil \log n, \text{ as } k\log b \leq 2\log n,\\
&\leq 2\frac{kn}{b}\log n + 6\log n\\
&\leq 2\frac{kn}{\left(\frac{n}{4\delta^k}\right)^{1/\lceil k/2\rceil} - 1}\log n + 6\log n\\
&= 2kn\frac{\delta^{k/\lceil k/2\rceil}}{n^{1/\lceil k/2\rceil}}\frac{1}{(1/4)^{1/\lceil k/2\rceil} - (\delta^k/n)^{1/\lceil k/2\rceil}}\log n + 6\log n\\
&\leq 2kn\frac{\delta^{k/\lceil k/2\rceil}}{n^{1/\lceil k/2\rceil}}8\lceil k/2\rceil\log n + 6\log n,\\
&\leq 16k^2 n^{1-1/\lceil k/2\rceil}2^{2M}\log n + 6\log n,
\end{aligned}$$

where the last inequality follows from $\delta \leq 2^M$, and the second-to-last inequality follows from combining this inequality with the assumption in the theorem that $n \geq 8\cdot 2^{kM}$ to show $n \geq 8\delta^k$, and applying the bound $(4^{-x} - 8^{-x})/x \geq 1/8$ for all $0 \leq x \leq 1$. ■

Lemma 2 *If $\mathcal{H}$ contains an ℓ-forbidden sub-hypergraph, then there is an input sequence of weight at most 2ℓ whose corresponding codeword in $C_{k,\pi,Q}$ has weight at most $2\ell + \ell\lceil kn/b\rceil$.*

Proof: Let S denote the set of edges of the ℓ-forbidden sub-hypergraph in $\mathcal{H}$, and consider the set B of bits of the input that correspond to the hyperedges of S. By definition, $B \subseteq U$ and $|B| \leq 2\ell$. We construct an input x of weight at most 2ℓ by setting the bits in B to 1 and other bits to 0, and consider the codeword corresponding to x: $(x, Q(\pi(r_k(x))))$. As each vertex of $\mathcal{H}$ is contained in an even number of the hyperedges in S, each interval in $\mathcal{I}$ contains an even number of bits that are 1 in $\pi(r_k(x))$. Thus, by the definition of U and Lemma 1, $Q(\pi(r_k(x)))$ is zero everywhere except inside those

intervals of $\mathcal{I}$ that contain a bit that is 1 in $\pi(r_k(x))$. Since there are at most ℓ such intervals, the weight of $Q(\pi(r_k(x)))$ is at most $\ell\lceil kn/b\rceil$. Therefore, the weight of the codeword corresponding to x is at most $2\ell + \ell\lceil kn/b\rceil$. ■

Lemma 3 *Every k-partite k-uniform hypergraph $\mathcal{H}$ with b vertices in each part and at least $4b^{\lceil k/2\rceil}$ hyper-edges contains a $k\log b$-forbidden sub-hypergraph.*

Proof: We construct a bipartite graph G from H as follows: For every $\lceil k/2\rceil$-tuple $(i_1, i_2, ..., i_{\lceil k/2\rceil})$ where i_j is a vertex in the j'th part of $\mathcal{H}$, we put a vertex in the first part of G, and for every $\lfloor k/2\rfloor$-tuple $(i_{\lceil k/2\rceil+1}, \ldots, i_k)$ where i_j is a vertex in the j'th part of $\mathcal{H}$, we put a vertex in the second part of G. If there is a hyperedge $\{i_1, i_2, \ldots, i_k\}$ in $\mathcal{H}$, where i_j is a vertex of the j'th part, we connect the vertices $(i_1, i_2, ..., i_{\lceil k/2\rceil})$ and $(i_{\lceil k/2\rceil+1}, \ldots, i_k)$ in G.

By the above construction, each edge in G corresponds to a hyperedge in $\mathcal{H}$. There are at least $4b^{\lceil k/2\rceil}$ edges and at most $2b^{\lceil k/2\rceil}$ vertices in G. Thus, by Lemma 4 below, G has a cycle of length at most $2\log(2b^{\lceil k/2\rceil}) < 2k\log b$. It is easy to see that the hyper-edges corresponding to the edges of this cycle constitute a $k\log b$-forbidden sub-hypergraph in $\mathcal{H}$. ■

Lemma 4 *Let G be a graph on n vertices with at least $2n$ edges. Then, G has a cycle of length at most $2\log n$.*

Proof: We first prove the theorem in the case that every vertex of G has degree at least 3. In this case, if the shortest cycle in the graph had length $2d+1$, then a breadth-first search tree of depth d from any vertex of the graph would contain at least $1 + 3\sum_{i=0}^{d-1} 2^i = 3\cdot 2^d - 2$ distinct vertices. As $3\cdot 2^{\log n} - 2 > n$, this would be a contradiction for $d \geq \log n$. So, the graph must contain a cycle of length at most $2\log n$.

We may prove the lemma in general by induction on n. Assume the lemma has been proved for all graphs with fewer than n vertices, and let G be a graph on n vertices with at least $2n$ edges. If the degree of every node in G is at least 3, then G has a cycle of length at most $2\log n$ by the preceding argument. On the other hand, if G has a vertex of degree 2, we consider the graph G' obtained by deleting this vertex and its two adjacent edges. The graph G' has $n-1$ vertices and at least $2(n-1)$ edges, and so by induction has a cycle of length at most $2\log(n-1)$. As G' is a subgraph of G, G also has a cycle of length at most $2\log(n-1) \leq 2\log n$, which proves the lemma. ■

2.3 Interleaver design for the parallel concatenated accumulator codes with two branches

The two-branch case of Theorem 2 implies that the minimum distance of a Turbo code of length n is at most $O(\log n)$. This fact was first proved in [4]. In this section, we show how to construct Turbo codes of message length n and minimum distance at least $\frac{1}{2}\log n$. The codes will be the parallel concatenation of accumulator codes. The key to our construction is an interleaver design technique inspired by Gallager's technique for designing sparse graphs without short cycles [6].

For a convolutional code Q, a positive integer n, and two permutations π_1 and π_2 of length n, let $C_{\pi_1,\pi_2,Q}$ denote the Turbo code formed by the parallel concatenation of Q with interleavers π_1 and π_2. That is, for any $x \in \{0,1\}^n$, $C_{\pi_1,\pi_2,Q}(x) = (x, Q(\pi_1(x)), Q(\pi_2(x)))$. While we may assume that π_1 is the identity permutation without loss of generality, we do not make use of this fact in this section.

For our construction, we let Q be an accumulator. We now construct a colored multigraph G_{π_1,π_2} with $n+1$ vertices from the permutations π_1 and π_2 as follows: join vertices $\pi_1(i)$ and $\pi_1(i+1)$ by a red edge for each $1 \leq i \leq n-1$, join vertices $\pi_2(i)$ and $\pi_2(i+1)$ by a blue edge for each $1 \leq i \leq n-1$, and connect $\pi_1(n)$ to n+1 by a red edge and connect $\pi_2(n)$ to n+1 by a blue edge. By our construction, G_{π_1,π_2} is a union of two paths of length n (*i.e.* having n edges). Similarly, from any graph G on $n+1$ vertices formed from the union of two colored paths that both end at vertex $n+1$, it is possible to construct permutations π_1 and π_2 such that $G = G_{\pi_1,\pi_2}$.

We will now prove Lemma 5, which shows that the minimum distance of $C_{\pi_1,\pi_2,Q}$ is at least the length of the shortest cycle in G_{π_1,π_2} (also known as the girth of G_{π_1,π_2}). Before reading the proof, the reader might want to verify that if the red path and blue path overlap, in which case there are a pair of vertices connected by both a red and blue edge, then graph has a cycle of length 2 and $C_{\pi_1,\pi_2,Q}$ has a word of weight 4: on input the vector x that is 1 at the bit positions indexed by these two vertices and zero elsewhere, both $Q(\pi_1(x))$ and $Q(\pi_1(x))$ have weight 1.

Lemma 5 *Let Q be an accumulator, n a positive integer, and π_1 and π_2 be two permutations of length n. The minimum weight of $(Q(\pi_1(x)), Q(\pi_2(x))$, for $x \in \{0,1\}^n$ is at least the length of the shortest cycle in G_{π_1,π_2} (irrespective of colors).*

Proof: For an $x \in \{0,1\}^n$, let $G_{\pi_1,\pi_2}(x)$ denote the subgraph of G_{π_1,π_2} including only those red edges $(\pi_1(i), \pi_1(i+1))$ for which $Q(\pi_1(x))_i = 1$ and those blue edges $(\pi_2(i), \pi_2(i+1))$ for which $Q(\pi_2(x))_i = 1$, where we extend the permutations so that

$\pi_1(n+1) = \pi_2(n+1) = n+1$. By Lemma 6, we know that every node in $G_{\pi_1,\pi_2}(x)$ has even degree. Thus, for non-zero x the graph must contain a cycle. ■

We remark that the minimum weight is actually equal to the length of the shortest cycle, and is obtained by setting x to be 1 on vertices between different colored edges in that cycle, and 0 elsewhere.

Lemma 6 *For every $x \in \{0,1\}^n$, every vertex of $G_{\pi_1,\pi_2}(x)$ has even degree.*

Proof: Let k be a vertex of G_{π_1,π_2} other than $n+1$. Let i and j be such that $\pi_1(i) = \pi_2(j) = k$. For vertex k, the claim follows from the facts that if $x_k = 0$ then $Q(\pi_1(x))_{i-1} = Q(\pi_1(x))_i$ and $Q(\pi_2(x))_{j-1} = Q(\pi_2(x))_j$, while if if $x_k = 1$ then $Q(\pi_1(x))_{i-1} + Q(\pi_1(x))_i = 1$ and $Q(\pi_2(x))_{j-1} + Q(\pi_2(x))_j = 1$. For vertex $n+1$, the lemma follows from the fact that both $Q(\pi_1(x))_n$ and $Q(\pi_2(x))_n$ equal are the sum modulo two of the bits in x. ■

Our code construction will follow from the following graph construction:

Lemma 7 *There is a polynomial-time algorithm that, on input $n \geq 5621$, outputs a graph G with $n+1$ vertices and girth $\frac{1}{2}\log n$ that is a disjoint union of two paths of length n both of which end at vertex $n+1$.*

Proof: We prove the following stronger result: For n sufficiently large, there is a graph G with $n+1$ vertices and girth at least $\frac{1}{2}\log n$ that is a disjoint union of two Hamiltonian cycles, i.e., two simple cycles of length $n+1$. One may obtain the desired paths from this graph by removing one edge from each cycle touching node $n+1$.

In order to construct such a graph G, our algorithm will start with an arbitrary graph G that is a disjoint union of two Hamiltonian cycles, one of them red and the other blue. The algorithm will then apply the following steps to change the blue Hamiltonian cycle until the girth of G becomes at least $\frac{1}{2}\log n$.

1. Let C be the shortest cycle in G. If the length of C is at least $\frac{1}{2}\log n$, then stop.
2. Consider a blue edge u_1u_2 in C. (Such an edge must exist, since the all-red cycle in G has length $n+1$).
3. Find a blue edge v_1v_2 in G such that

$$\mathsf{dist}(u_i, v_j) > \frac{1}{2}\log n \quad \text{for every } i = 1,2 \text{ and } j = 1,2. \tag{2}$$

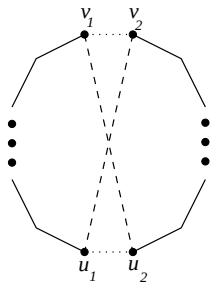

Figure 1: Switching two edges of the blue Hamiltonian cycle

Here $\mathsf{dist}(u, v)$ denotes the length of the shortest path in G between the vertices u and v.

4. Remove the edges u_1u_2 and v_1v_2 from G, and instead, add either the blue edges u_1v_1 and u_2v_2, or the blue edges u_1v_2 and u_2v_1 so that the set of blue edges remains a Hamiltonian cycle. (See Figure 1).

5. Go to step 1 and repeat.

To establish the correctness of this algorithm, we must prove that the edge v_1v_2 can always be found in step 3, and that the algorithm always terminates. We first show that, assuming $n \geq 5621$, the edge v_1v_2 can be found in step 3. The number of nodes at distance at most $\frac{1}{2}\log n$ from u_i is at most $2 \cdot 3^{\frac{1}{2}\log n}$. As u_1 is connected to u_2, we similarly find that the number of vertices at distance at most $\frac{1}{2}\log n$ from either u_1 or u_2 is at most $3 \cdot 3^{\frac{1}{2}\log n}$. As there are $n+1$ blue edges, each of the $3 \cdot 3^{\frac{1}{2}\log n}$ vertices can account for at most 2 edges, and $6 \cdot 3^{\frac{1}{2}\log n} < n$ for $n \geq 5621$, the desired blue edge v_1v_2 must exist.

To show that the algorithm converges, we will prove that, at each iteration, either the length of the shortest cycle increases, or that the number of edges in shortest-length cycles decreases. This will imply that the algorithm terminates within at most $O(n \log n)$ iterations. It suffices to show that every cycle in G involving one of the newly inserted edges must be longer than the cycle that was broken. This follows from property (2), which implies that each cycle of length at most $\frac{1}{2}\log n$ in the graph G after step 4 that contains one of the newly inserted edges must consist of both of these edges, a path between u_1 and u_2, and a path between v_1 and v_2. However, such a cycle must have length greater than C.

To prove that the algorithm can be implemented in polynomial time, we note that the shortest cycle involving a given vertex, or the set of vertices within a given distance of

another, may be found in $O(n)$ time. Finally, it is obvious that when the algorithm terminates, the graph G is a disjoint union of two Hamiltonian cycles and the girth of G is at least $\frac{1}{2}\log n$. ■

Consequently:

Theorem 3 *Let Q be an accumulator, and $n \geq 5621$ denote the message length. There exist polynomial time constructible permutations π_1, π_2 such that the minimum distance of the Turbo code $C_{\pi_1,\pi_2,Q}$ is at least $\frac{1}{2}\log n$.*

Proof: By Lemma 7, we can construct a graph G on $n+1$ vertices of girth at least $\frac{1}{2}\log n$ that is a disjoint union of two Hamiltonian paths. We know that corresponding to such a graph G, we can find permutations π_1 and π_2 such that $G = G_{\pi_1,\pi_2}$. By Lemma 5, the minimum distance of the code $C_{\pi_1,\pi_2,Q}$ is equal to the girth of G, which is at least $\frac{1}{2}\log n$. ■

3 Serially concatenated Turbo-like codes

In this section, we consider codes that are obtained by serially concatenating convolutional codes and, more generally, automata codes. In Section 3.1, we prove an upper bound on the minimum distance of the concatenation of a low-memory outer automata encoder with an arbitrary inner automata encoder. In particular, we prove that if the memory of the outer code is constant and the memory of the inner code is sub-linear, then the code is asymptotically bad. In contrast, in Section 3.2, we prove that if the input is first passed through a repetition code and a random permutation, then the code is asymptotically good with constant probability, even if both convolutional encoders are accumulators.

3.1 Upper bound on the minimum distance when the outer code is weak

In this section, we consider the serial concatenation of automata codes. We assume that each automata outputs a constant number of bits per transition. This class of codes includes the standard serially concatenated Turbo codes, and includes those introduced by Benedetto, Divsalar, Montorsi and Pollara [2] and studied by Kahale and Urbanke [8]. If the outer code has constant memory and the inner code has sub-linear memory, then our bound implies that the code cannot be asymptotically good.

Formally, we assume that Q_o (Q_i, respectively) is an automata encoder with at most 2^{M_o} (2^{M_i}, respectively) states and h_o (h_i, respectively) output bits per transition. For an integer n and a permutation π of length $h_o n$, we define $C_{Q_o,Q_i,\pi}$ to be the code that encodes a string $x \in \{0,1\}^n$ to the string $C_{Q_o,Q_i,\pi}(x) := Q_i(\pi(Q_o(x))) \in \{0,1\}^{h_o h_i n}$. We will assume without loss of generality that Q_o, Q_i, and π are such that this mapping is an injective mapping. The encoders Q_o and Q_i are called the outer and inner encoders, respectively.

Theorem 4 *Let Q_o be an automata encoder with at most 2^{M_o} states that outputs h_o bits at each time step, and let Q_i be an automata encoder with at most 2^{M_i} states that outputs h_i bits at each time step. For any positive integer n and any permutation π of length nh_o, the minimum distance of the code $C_{Q_o,Q_i,\pi}$ is at most*

$$3h_o^2 h_i (M_o + 2) n^{1 - \frac{1}{h_o(M_o+2)}} M_i^{\frac{1}{h_o(M_o+2)}}.$$

In particular, if M_o is constant (and h_i and h_0 are constants), the minimum distance of the code $C_{Q_o,Q_i,\pi}$ is

$$O(n^{1 - \frac{1}{h_o(M_o+2)}} M_i^{\frac{1}{h_o(M_o+2)}}),$$

and consequently any such family of codes $C_{k,\pi,Q}$ is asymptotically bad as long as M_i is sub-linear in n.

Proof: The proof follows the same outline as the proof of Theorem 1. We begin by setting $I_o = \{1, \ldots, n\}$ to be the set of times steps in the computation of Q_o on input $x \in \{0,1\}^n$, and setting $I_i = \{1, \ldots, h_o n\}$ to be the set of times steps in the computation of Q_i on input $\pi\left(Q_o(x)\right) \in \{0,1\}^{h_o n}$. We similarly, let $\left\{s_o^{(t)}(x)\right\}_{t \in I_o}$ denote the sequence of states traversed by Q_o on input x and $\left\{s_i^{(t)}(x)\right\}_{t \in I_i}$ denote the sequence of states traversed by Q_i on input $\pi\left(Q_o(x)\right)$.

To prove the claimed bound on the minimum distance of $C_{Q_o,Q_i,\pi}$, we will prove the existence of two distinct input strings x and y, a set $V \subset \{1, \ldots, n\}$, a set $J_o \subset I_o$, and a set $J_i \subset I_i$ such that x and y are both 0 on bits not in V, $s_o^{(t)}(x)$ and $s_o^{(t)}(y)$ only differ for $t \in J_o$, and $s_i^{(t)}(x)$ and $s_i^{(t)}(y)$ only differ for $t \in J_i$. The minimum distance bound will then follow from an upper bound on the size of J_i.

To construct these sets, we make use of parameters m_o and m_i to be determined later. We first partition the set I_o into $b_o \stackrel{\text{def}}{=} \lfloor n/m_o \rfloor$ intervals each of size m_o or $m_o + 1$, and we partition the set I_i into $b_i \stackrel{\text{def}}{=} \lfloor nh_o/m_i \rfloor$ intervals each of size m_i or $m_i + 1$.

As Q_o outputs at most $(m_o+1)h_o$ bits during the time steps in an interval in I_o, the bits output by Q_o during an interval in I_o are read by Q_i during at most $(m_o+1)h_o$ intervals in I_i. As there are fewer than $(b_i)^{(m_o+1)h_o}$ sets of at most $(m_o+1)h_o$ intervals in I_i, there exists a set of at least $b_o/(b_i)^{(m_o+1)h_o}$ intervals in I_o such that all the bits output by Q_o during these intervals are read by Q_i during a single set of at most $(m_o+1)h_o$ intervals in I_i. Let U denote the set of at least $b_o/(b_i)^{(m_o+1)h_o}$ intervals in I_o and let T denote the corresponding set of at most $(m_o+1)h_o$ intervals in I_i. We then let V denote the set of input bits read by Q_o during the intervals in U. As all the intervals in I_o have size at least m_o, we have $|V| \geq m_o\,|U|$. The set J^o will be the union of the intervals in T and J^i will be the union of the intervals in U.

Let $\{u_j\}_{j=1}^{|U|}$ and $\{t_j\}_{j=1}^{|T|}$ denote the last time steps in the intervals in U and T respectively. For each $x \in \{0,1\}^n$ that is zero outside V, we consider $\left(s_o^{(u_j)}(x)\right)_{j=1}^{|U|}$, the sequence of states traversed by Q_o on x at times $u_1, \ldots, u_{|U|}$, and, $\left(s_i^{(t_j)}(x)\right)_{j=1}^{|T|}$, the sequence of states traversed by Q_i on input $\pi(Q_o(x))$ at times $t_1, \ldots, t_{|T|}$. There are at most $2^{M_o|U|}2^{M_i|T|}$ such pairs of sequences. So, if

$$2^{M_o|U|}2^{M_i|T|} < 2^{|V|}, \tag{3}$$

then there are two distinct x and y in $\{0,1\}^n$ that are both 0 outside V and a pair of sequences $\left(s_i^{(t_j)}\right)_{j=1}^{|T|}$ and $\left(s_o^{(u_j)}\right)_{j=1}^{|U|}$ such that $s_i^{(t_j)}(x) = s_o^{(t_j)}(y) = s_i^{(t_j)}$ for all $1 \leq j \leq |T|$ and $s_o^{(u_j)}(x) = s_o^{(u_j)}(y) = s_o^{(u_j)}$ for all $1 \leq j \leq |U|$. This means that the bits output and states traversed by Q_o on inputs x and y are the same at time steps outside the time intervals in U, and therefore the bits output and states traversed by Q_i on inputs $\pi(Q_o(x))$ and $\pi(Q_o(y))$ are the same outside time steps in intervals in T. Thus

$$0 < d(C_{Q_i,Q_o,\pi}(x), C_{Q_i,Q_o,\pi}(y)) \leq m_i h_i\,|T| \leq (m_o+1)m_i h_o h_i. \tag{4}$$

As this bound assumes (3), we will now show that for

$$\begin{aligned} m_o &= M_o + 1, \text{ and} \\ m_i &= 3h_o n^{1-\frac{1}{(M_o+2)h_o}} (M_i)^{\frac{1}{(M_o+2)h_o}}, \end{aligned}$$

this assumption is true.

Our setting of m_o reduces (3) to

$$|U| \geq |T|\,M_i,$$

which would be implied by

$$\frac{b_o}{b_i^{(m_o+1)h_o}} > (m_o+1)h_o M_i. \tag{5}$$

To derive this inequality, we first note that since $x^{2/x} < 3$ for $x \geq 1$,

$$m_i > h_o n^{1-\frac{1}{(M_o+2)h_o}} (((m_o+1)h_o)^2 M_i)^{\frac{1}{(M_o+2)h_o}}.$$

Rearranging terms, we find this implies

$$\left(\frac{n}{(m_o+1)^2 h_o^2 M_i}\right)^{\frac{1}{(m_o+1)h_o}} > \frac{nh_o}{m_i} \geq b_i.$$

Again rearranging terms, we obtain

$$n > b_i^{(m_o+1)h_o}(m_o+1)^2 h_o^2 M_i \geq b_i^{(m_o+1)h_o}(m_o+1)m_o h_o M_i + m_o,$$

which implies

$$\left\lfloor \frac{n}{m_o} \right\rfloor > b_i^{(m_o+1)h_o}(m_o+1)h_o M_i.$$

By now dividing both sides by $b_i^{(m_o+1)h_o}$ and recalling $b_o = \left\lfloor \frac{n}{m_o} \right\rfloor$, we derive (5).

Finally, the bound on the minimum distance of the code now follows by substituting the chosen values for m_o and m_i into (4).

■

We now compare this with the high-probability upper bound of $\tilde{O}(n^{1-2/d_o} 2^{M_i})$ on the minimum distance of rate 1/4 random serially concatenated codes obtained by Kahale and Urbanke [8]. In their case, we have $h_o = h_i = 2$, and our upper bound becomes $O(n^{1-1/(2M_o+4)}\ M_i^{1/(2M_o+4)})$. We note that the dependence of our bound on d_o is comparable, and the dependence of our bound on M_i is much better.

3.2 A strong outer code: when serially concatenated Turbo-like codes become asymptotically good

The proof technique used in Theorem 4 fails if the outer code is not a convolution code or encodable by a small finite automata. This suggests that by strengthening the outer code one might be able to construct asymptotically good codes. In fact, we will

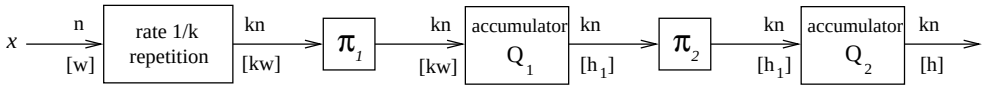

Figure 2: an RAA code

prove that the serial concatenation of an outer repeat-accumulate code with an inner accumulator yields an asymptotically good code with some positive probability.

Let $k \geq 2$ be an integer, r_k be the repeat-k-times map, Q_1 and Q_2 be accumulators[2], n be an integer, and π_1 and π_2 be permutations of length kn. We define C_{k,π_1,π_2} to be the code that maps input strings $x \in \{0,1\}^n$ to $C_{k,\pi_1,\pi_2}(x) := Q_2(\pi_2(Q_1(\pi_1(r_k(x)))))$. We call C_{k,π_1,π_2} an RAA (Repeat, Accumulate, and Accumulate) code (See Figure 2). We note that this code has rate $1/k$.

In contrast with the codes analyzed in Theorem 4, these RAA codes have a repeat-accumulate code, $C_{k,\pi_1}(y) = Q_1(\pi_1(r_k(x))$ where those analyzed in Theorem 4 merely have an automata encoder.

Theorem 5 *Let $k \geq 2$ and n be integers, and let π_1 and π_2 be permutations of length kn chosen uniformly at random. Then for each constant $\delta > 0$, there exists a constant $\epsilon > 0$ and an integer n_0, such that the RAA code C_{k,π_1,π_2} has minimum distance at least ϵn with probability at least $1 - \delta$ for all $n \geq n_0$.*

So specifically, there exists an infinite family of asymptotically good RRA codes.

Proof: Conditions bounding the size of ϵ will be appear throughout the proof.

Let $E_{\epsilon n}$ denote the expected number of non-zero codewords in C_{k,π_1,π_2} of weight less than or equal to ϵn. Taking a union bound over inputs and applying linearity of expectation, we see that the probability the minimum distance of C_{k,π_1,π_2} is less than ϵn is at most $E_{\epsilon n}$. Thus, we will bound this probability by bounding $E_{\epsilon n}$.

To bound $E_{\epsilon n}$, we use techniques introduced by Divsalar, Jin and McEliece [5] for computing the expected input-output weight enumerator of random Turbo-like codes. For an accumulator code of message length N, let $A_{w,h}^{(N)}$ denote the number of inputs of weight w on which the output of the accumulator has weight h. Divsalar, Jin and McEliece [5] prove that

[2]While Q_1 and Q_2 are identical as codes, we give them different names to indicate their different roles in the construction.

$$A_{w,h}^{(N)} = \binom{N-h}{\lfloor w/2 \rfloor}\binom{h-1}{\lceil w/2 \rceil - 1}, \tag{6}$$

where $\binom{a}{b}$ is defined to be zero if $a < b$. Therefore, if the input to Q is a random string of length N and weight w, the probability that the output has weight h is

$$\frac{A_{w,h}^{(N)}}{\binom{N}{w}} = \frac{\binom{N-h}{\lfloor w/2 \rfloor}\binom{h-1}{\lceil w/2 \rceil - 1}}{\binom{N}{w}}. \tag{7}$$

Now consider a fixed input x to the encoder for C_{k,π_1,π_2}. If x has length n and weight w and π_1 is a random permutation of length kn, then $\pi_1(r_k(x))$ is a random string of length kn and weight kw. This random string is the input to the accumulator Q_1. Therefore, by (7), for any h_1 the probability that the output of Q_1 has weight h_1 is $A_{kw,h_1}^{(kn)}/\binom{kn}{kw}$. If this happens, the input to Q_2 will be a random string of weight h_1, and therefore, again by (7), the probability that the output of Q_2 has weight h will be equal to $A_{h_1,h}^{(kn)}/\binom{kn}{h_1}$. Thus, for any fixed input string x of weight w, and any fixed h_1 and h, the probability over the choice of π_1 and π_2 that the output of Q_1 has weight h_1 and the output of Q_2 (which is also the output of C_{k,π_1,π_2}) has weight h is equal to

$$\frac{A_{kw,h_1}^{(kn)} A_{h_1,h}^{(kn)}}{\binom{kn}{kw}\binom{kn}{h_1}}.$$

Thus, by the linearity of expectation, the expected number of non-zero codewords of C_{k,π_1,π_2} of weight at most ϵn equals

$$E_{\epsilon n} = \sum_{w=1}^{n}\sum_{h_1=0}^{kn}\sum_{h=1}^{\epsilon n}\frac{\binom{n}{w}A_{kw,h_1}^{(kn)}A_{h_1,h}^{(kn)}}{\binom{kn}{kw}\binom{kn}{h_1}} = \sum_{h_1=1}^{2\epsilon n}\sum_{w=1}^{2h_1/k}\sum_{h=1}^{\epsilon n}\frac{\binom{n}{w}A_{kw,h_1}^{(kn)}A_{h_1,h}^{(kn)}}{\binom{kn}{kw}\binom{kn}{h_1}},$$

as the terms with $\lceil h_1/2 \rceil > h$ or $\lceil kw/2 \rceil > h_1$ are zero. Using the inequalities $\binom{x}{y} \leq (ex/y)^y$, $\binom{x}{\lfloor y/2 \rfloor} \leq (4ex/y)^y$ and $\binom{x}{\lceil y/2 \rceil - 1} \leq (4ex/y)^y$, for positive integers x and y, we

bound this sum by

$$\begin{aligned}
E_{\epsilon n} &= \sum_{h_1=1}^{2\epsilon n}\sum_{w=1}^{2h_1/k}\sum_{h=1}^{\epsilon n}\frac{\binom{n}{w}\binom{kn-h_1}{\lfloor kw/2\rfloor}\binom{h_1-1}{\lceil kw/2\rceil-1}\binom{kn-h}{\lfloor h_1/2\rfloor}\binom{h-1}{\lceil h_1/2\rceil-1}}{\binom{kn}{kw}\binom{kn}{h_1}} \\
&\leq \sum_{h_1=1}^{2\epsilon n}\sum_{w=1}^{2h_1/k}\sum_{h=1}^{\epsilon n}\frac{\binom{n}{w}\left(\frac{4ekn}{kw}\right)^{kw/2}\left(\frac{4eh_1}{kw}\right)^{kw/2}\left(\frac{4ekn}{h_1}\right)^{\lfloor h_1/2\rfloor}\left(\frac{4eh}{h_1}\right)^{\lceil h_1/2\rceil-1}}{\left(\frac{n}{w}\right)^{kw}\left(\frac{kn}{h_1}\right)^{h_1}} \\
&= \sum_{h_1=1}^{2\epsilon n}\sum_{w=1}^{2h_1/k}\sum_{h=1}^{\epsilon n}\binom{n}{w}\left(\frac{4e\sqrt{h_1}}{\sqrt{kn}}\right)^{kw}\left(\frac{h}{kn}\right)^{\lceil h_1/2\rceil}\frac{(4e)^{h_1-1}h_1}{h}
\end{aligned}$$

The summand in the above expression is at maximum when $h=\epsilon n$. Therefore,

$$\begin{aligned}
E_{\epsilon n} &\leq \epsilon n\sum_{h_1=1}^{2\epsilon n}\left(\frac{\epsilon n}{kn}\right)^{\lceil h_1/2\rceil}\frac{h_1(4e)^{h_1-1}}{\epsilon n}\sum_{w=1}^{2h_1/k}\binom{n}{w}\left(\frac{4e\sqrt{h_1}}{k\sqrt{n}}\right)^{kw} \\
&\leq \sum_{h_1=1}^{2\epsilon n}h_1\left(4e\sqrt{\epsilon/k}\right)^{h_1}\sum_{w=1}^{2h_1/k}\binom{n}{w}\left(\frac{4e\sqrt{h_1}}{k\sqrt{n}}\right)^{kw} \\
&\leq \sum_{h_1=1}^{2\epsilon n}h_1\left(4e\sqrt{\epsilon/k}\right)^{h_1}\sum_{w=1}^{2h_1/k}\left(\frac{ne}{w}\right)^{w}\left(\frac{4e\sqrt{h_1}}{k\sqrt{n}}\right)^{kw} \\
&= \sum_{h_1=1}^{2\epsilon n}h_1\left(4e\sqrt{\epsilon/k}\right)^{h_1}\sum_{w=1}^{2h_1/k}\left(\frac{e\left(\frac{4e}{k}\right)^{k}n^{1-k/2}h_1^{k/2}}{w}\right)^{w} \\
&\leq \sum_{h_1=1}^{2\epsilon n}h_1\left(4e\sqrt{\epsilon/k}\right)^{h_1}\frac{2h_1}{k}e^{\left(\frac{4e}{k}\right)^{k}n^{1-k/2}h_1^{k/2}}, \text{ as } \left(\frac{y}{x}\right)^{x}\leq e^{y/e} \\
&\leq \frac{2}{k}\sum_{h_1=1}^{2\epsilon n}\left(4e^2\sqrt{\epsilon/k}\right)^{h_1}e^{\left(\left(\frac{4e^2}{k}\right)^{k}n^{1-k/2}\right)h_1^{k/2}}, \qquad (8)
\end{aligned}$$

since $h_1^2\leq e^{h_1}$ for all $h_1\geq 1$. To bound (8), note that the sum has the form

$$S=\sum_{x=1}^{m}\alpha^x e^{\beta x^l},$$

where $\alpha=4e^2\sqrt{\epsilon/k}$, $\beta=\left(\frac{4e^2}{k}\right)^{k}n^{1-k/2}$, $l=\frac{k}{2}$, and $m=2\epsilon n$. If we can guarantee that

$$\alpha^{x+1}e^{\beta(x+1)^l}\leq\frac{1}{2}\alpha^x e^{\beta x^l}, \qquad (9)$$

for all $x = 1, \ldots, m-1$, we can use the bound

$$S \leq 2\alpha e^{\beta}. \tag{10}$$

We can express (9) as $\beta((x+1)^l - x^l) \leq \ln \frac{1}{2\alpha}$. Thus (9) holds for all the desired values of x if $\beta((m+1)^l - m^l) \leq \ln \frac{1}{2\alpha}$, or equivalently

$$\beta m^l \left(\left(1 + \frac{1}{m} \right)^l - 1 \right) \leq \ln \frac{1}{2\alpha},$$

which can be guaranteed when

$$2l\beta m^{l-1} \leq \ln \frac{1}{2\alpha} \quad \text{and} \quad l \leq m, \tag{11}$$

via the bounds

$$\left(1 + \frac{1}{m} \right)^l \leq e^{l/m} \leq 1 + (e-1)\frac{l}{m} \leq 1 + 2\frac{l}{m},$$

where we need $l \leq m$ in the second inequality. Going back to (8), we get via (10) and (11) that

$$E_{\epsilon n} \leq \frac{2}{k} 2(4e^2\sqrt{\epsilon/k}) e^{\left(\frac{4e^2}{k}\right)^k n^{1-k/2}} = \frac{16e^2\sqrt{\epsilon}}{k\sqrt{k}} e^{\left(\frac{4e^2}{k}\right)^k n^{1-k/2}}, \tag{12}$$

when

$$2\frac{k}{2}\left(\frac{4e^2}{k}\right)^k n^{1-k/2}(2\epsilon n)^{k/2-1} \leq \ln\left(\frac{1}{8e^2}\sqrt{\frac{k}{\epsilon}}\right) \quad \text{and} \quad \frac{k}{2} \leq 2\epsilon n,$$

or, equivalently, when

$$\ln \frac{1}{\epsilon} \geq \left(2k \left(\frac{4e}{k} \right)^k 2^{k/2-1} \right) \epsilon^{k/2-1} - 2\ln \frac{\sqrt{k}}{8e^2} \quad \text{and} \quad \frac{k}{2} \leq 2\epsilon n. \tag{13}$$

It follows from (12) and (13), that for each $k \geq 2$, and for each constant $\delta > 0$, there is constant $\epsilon > 0$ such $E_{\epsilon n} < \delta$ when n is sufficiently large.

■

While the constants we obtain are not particularly sharp, they are sufficient to prove the existence of asymptotically good families of serially concatenated turbo-like codes of depth 2. This result should be compared with the work of Pfister and Siegel [9], who performed experimental analyses of the serial concatenation of repetition codes with l levels of accumulators connected by random interleavers, and theoretical analyses of of concatenations of a repetition code with certain rate-1 codes for large l. Their experimental results indicate that the average minimum distance of the ensemble starts becoming good for $l \geq 2$, which is consistent with our theorem. For certain rate-1 codes and l going to infinity, they proved their codes could become asymptotically good. In contrast, we prove this for $l = 2$ and accumulator codes.

4 Open questions

- Can the RAA codes described in Section 3.2 be efficiently decoded by iterative decoding, or any other algorithm?
- Can one obtain depth-3 serially concatenated codes with better minimum distance by replacing the accumulators in the RAA codes with small convolutional codes? Also, can one improve the minimum distance bounds on the RAA codes?
- If one allows the memory of the outer code in a depth-2 serially concatenated code to grow logarithmically with the block length, can one obtain an asymptotically good code?
- Can one improve upon the interleaver construction in Section 2.3 so that it applies when the constituent codes are more complicated than accumulators? Also, can one obtain algebraically explicit interleavers rather than an algorithmic construction.
- Can one construct interleavers for parallel concatenated Turbo codes with more than two branches that result in a minimum distance close to that obtained with random interleavers?
- Is it possible to interpolate between the bounds of Theorems 1 and 2?

5 Acknowledgment

We thank Sanjoy Mitter for many stimulating conversations.

References

[1] M. Ajtai. Determinism versus non-determinism for linear time RAMs with memory restrictions. In *Proc. of the 31st Ann. ACM Symp. on Theory of Computing (STOC)*, pages 632–641, 1999.

[2] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding. In *IEEE Transactions on Information Theory, vol. 44 (3), pages 909–926, May 1998.*

[3] *C. Berrou, A. Glavieux, and P. Thitimajashima. Near Shannon limit error-correction coding: Turbo Codes. In* Proc. 1993 IEEE International Conference on Communications, *pages 1064–1070, Geneva, Switzerland, May 1993.*

[4] *M. Breiling. A Logarithmic Upper Bound on the Minimum Distance of Turbo Codes. submitted to the IEEE Transactions on Information Theory.*

[5] *D. Divsalar, H. Jin, and R. J. McEliece. Coding theorems for "turbo-like" codes. In* Proc. 36th Annual Allerton Conference on Comm., Control, and Computing, *pages 210–210, Sept. 1998.*

[6] *R. G. Gallager.* Low Density Parity Check Codes. *Monograph, M.I.T. Press, 1963.*

[7] *H. Jin and R.J. McEliece. RA Codes Achieve AWGN Channel Capacity. In* Proc. 13th International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes, *volume 1719 of* Lecture Notes in Computer Science, *pages 10–18, 1999.*

[8] *N. Kahale and R. Urbanke. On the Minimum Distance of Parallel and Serially Concatenated Codes. to appear in IEEE Transactions on Information Theory, 1997.*

[9] *H. D. Pfister and P. H. Siegel. The Serial Concatenation of Rate-1 Codes Through Uniform Random Interleavers. In* Proc. 37th Allerton Conference on Communication, Control and Computing, *pages 260–269, Monticello, Illinois, Sep 1999.*

[10] *B. Vucetic and J. Yuan.* Turbo Codes: Principles and Applications. *Kluwer Academic Publishers, 2000.*