Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 42 (2011)

Efficiently Coding for Interactive Communication

Ankur Moitra [*]
MIT
moitra@mit.edu

Abstract

In 1992, Schulman [11] proved a coding theorem for interactive communication and demonstrated that interactive communication protocols can be made robust to noise with only a constant slow-down (for a sufficiently small error rate) through a black-box reduction. This protocol fails over a noisy channel with only exponentially small probability. However, this scheme is not computationally *efficient*: the running time to construct a good distance *tree code* (and perform encoding and decoding), which is the basis for the simulation, requires time exponential in the length of the protocol. Here, we give a computationally efficient simulation that achieves constant slow-down, and fails over a noisy channel with only polynomially small probability. Our protocol is deterministic and is based on a new type of code, which we call a *local tree code*. These codes can be regarded as an embedding of a tree code into a high-girth expander, so that locally these codes resemble tree codes, but are concisely represented and admit efficient encoding and decoding schemes (that succeed with high probability when communicating over a noisy channel).

[*]This research was supported in part by a Fannie and John Hertz Foundation Fellowship.

ISSN 1433-8092

1 Introduction

1.1 Background

In a landmark paper in 1948, Shannon initiated the study of error correcting codes [14]. Shannon demonstrated that a packet of k bits can be encoded into an $O(k)$ bit message so that even if each bit in this message is corrupted with some independent probability, the original k bit message can be recovered (and this procedure fails with exponentially small probability). The improvement over the naive protocol is dramatic - in order to achieve exponentially small error probability, the naive retransmission protocol needs to send k^2 bits. However, Shannon's proof did not yield an error correcting code that is computationally efficient (for which both encoding and decoding run in time polynomial in k). Today, a variety of efficiently encodable and decodable error correcting codes are known.

In 1992, Schulman [11] proved a coding theorem for interactive communication - in this setting, two parties – processor A and processor B receive pieces of the input (x and y respectively), and the goal is to compute some function of the pair of inputs (x, y). For an introduction to communication complexity see [9] or [17]. The naive protocol in this context is to send x from processor A to processor B (or vice-versa), but in many contexts the number of bits of communication can be substantially reduced. Schulman demonstrated that interactive communication protocols can be made robust to noise. Schulman did this through a blackbox reduction. We measure the cost of a protocol as the maximum number of bits needed to run the protocol on any pair of inputs, and we will denote this quantity by $|\pi|$. Given any protocol π, Schulman gave a method to simulate this protocol with only constant slow-down, while still guaranteeing a correct output – provided that the error probability η is sufficiently small. This protocol fails over a noisy channel with only exponentially small probability. In fact, Schulman's simulation works in a more general, adversarial context. This result has been recently improved by Braverman and Rao who improve the tolerable rate of error from $\frac{1}{240}$ to $\frac{1}{8} - \epsilon$ (for binary codes) [3].

However, the difficulty is that this coding scheme is not *efficient*, in the sense that the running time is exponential in the length $|\pi|$ of the noiseless protocol. This difficulty arises not only in the task of encoding or decoding, but even in fully specifying the simulation strategy (which is based on a *tree code*) that Schulman used in this coding theorem.

The challenge in efficiently coding for interactive communication is that we can regard the bits that one processor sends to the other processor over the duration of the protocol as a message (which is generated on-line) that needs to be encoded. In fact, if we could non-deterministically guess the path taken in the protocol π, then verifying that this path is indeed taken is exactly the setting considered by Shannon. A more subtle challenge is that, unlike the setting considered by Shannon in which a random code is almost surely a good code, a random *tree code* is almost surely not a good tree code (and hence does not have good error correction properties).

Here, we give an efficient simulation for interactive communication on a lossy channel which achieves a constant slowdown. Our protocol is based on a novel type of tree-like code – which we call a *local tree code* – based on a high-girth expander. These graphs appear to be locally tree-like. Moreover, in Schulman's original protocol only a polynomial sized region of a *tree code* is actually explored during a simulation. We do not know this region in advance, but we are able to embed this explored region into a size $O(|\pi|)$ high-girth expander. Also, our proof that good local tree codes exist is based on the general Lovász Local Lemma. This connection helps explain why a random tree code (or similarly a random local tree code) almost surely has zero distance, yet good tree codes and good local tree codes exist. We are then able to use the recent constructive proof of the general Lovász Local Lemma due to Moser and Tardos [10] (and improvements due to [5]) to efficiently construct good local tree codes. Here, however, our protocol fails with polynomially small probability (over a noisy channel) as opposed to exponentially small probability as in Schulman's protocol. Also, our results do not apply to the adversarial channel setting due to the bursty errors that an adversary could introduce.

In concurrent work, Gelles and Sahai introduced a relaxation of the notion of a tree code, called a *potent tree code*. Gelles and Sahai demonstrated that a randomly chosen potent tree code is "good" almost surely, and were able these codes to simulate interactive communication protocols (with constant slowdown) over both noisy and adversarial channels. In the case of a noisy channel, this protocol fails with only exponentially small probability and furthermore can be encoded in polynomial time and decoded in expected polynomial time. Given a "good" potent tree code, the protocol in [**?**] still succeeds over an adversarial channel but decoding is no longer efficient.

1.2 Our Results

Here we state our main result:

Theorem 1. *There is an alphabet Σ of size $O(1)$ and a transformation $L(\pi)$ – where π is a binary communication protocol – so that $|L(\pi)| = O(|\pi|)$, and for a channel noise rate η smaller than some universal constant, both parties can determine $\pi(x, y)$ by running $L(\pi)$ with high probability. This communication protocol runs in time polynomial in $|\pi|$.*

In fact, this simulation can be made deterministic, and still runs in time polynomial in $|\pi|$. In our proof, we will introduce the notion of a *local tree code*, prove that good local tree codes exist and can be efficiently computed (deterministically). We believe that these codes are of general interest, independent of the applications to efficiently coding for interactive communication. Understanding the limits of good local tree codes remains an interesting open question.

2 Notation

2.1 Model

A binary communication protocol π is a rooted tree where each internal vertex has four children, and the outgoing links to these children are mapped one-to-one to the set "00", "01", "10", and "11". Each internal node u is labelled by a pair of functions f_A^u and f_B^u, each from the respective domains of processor A and processor B to the set $\{0, 1\}$. When the communication protocol reaches node u, the protocol traverses the outgoing link specified by $f_A^u(x) \times f_B^u(y)$. We will denote $\pi(x, y)$ as the output of the communication protocol when x and y are the inputs to processor A and processor B respectively. Additionally, we will denote by $|\pi|$ the maximum length of the protocol on any pair of inputs.

Throughout this paper, we will focus on the *noisy channel* model, in which each message (a symbol from Σ) will be corrupted with probability η, and will remain intact with the remaining probability $1 - \eta$. Our results are independent of *how* a corrupted symbol is corrupted - i.e. is it resampled uniformly at random from Σ, or is it adversarially specified as some other symbol in Σ by an adversary. All we will require is that the probability of each individual message being corrupted is p, which will preclude certain types of error patterns during a (transformed) communication protocol. Also, we will let $\Delta(s, t)$ be the Hamming distance between two strings s and t of the same length.

2.2 Tree Codes

The notion of a tree code, as we present it, was introduced by Schulman in [13]:

Definition 1. *A d-ary tree code of depth n on an alphabet Σ is a tree of depth n whose internal nodes have d children, and each outgoing link from an internal node u is labeled by a (distinct) element of Σ*

The notion of the distance of a tree code captures the error-correcting properties of this structure. Given an internal node u, let $W(u) \in \Sigma^{depth(u)}$ denote the labels of edges traversed in the unique path from the root to u.

Definition 2. *A tree code has distance α if for all pairs of nodes u and v (suppose $depth(u) \leq depth(v)$), $\Delta(W(u)_{1,\dots depth(u)}, W(v)_{1,\dots depth(u)}) \leq \alpha(depth(u) - depth(w))$ where w is the least common ancestor of u and v.*

Schulman proved that tree codes with constant distance exist (for constant sized alphabets – good tree codes are easier to construct on larger alphabets). Note here that the probabilistic method cannot be naively applied, because simply choosing the symbols on each edge uniformly at random will not suffice. Consider two leaf nodes u and v for which $parent(u) = parent(v) = w$. The edges (w, u) and (w, v) must use different symbols. Yet if we were to choose symbols randomly for each edge in a tree code of depth n, there is a constant chance that this edge would be bad (and result in a tree code of distance 0) because the symbol on these edges is the same.

Schulman constructed these codes iteratively, from smaller and smaller depth tree codes. We will face a similar problem in constructing good distance (local variants) of tree codes, but to circumvent this problem we will apply the (general) Lovász Local Lemma – and we will appeal to the results of Moser and Tardos [10] to construct good distance (local) tree codes.

Since the introduction of tree codes in [13], no efficient tree codes have been given – i.e. there are no known tree codes of constant distance (and constant alphabet) that have efficient (polynomial in the *depth*) encoding and decoding schemes. The obstacle is that to even specify the tree code in this form requires specifying each symbol on each of the exponentially many in n edges.

The history of error correcting codes encountered, and overcame a similar problem: Shannon proved that good rate, good distance codes exist by choosing a random dictionary. This encoding scheme (mapping an input word to an element in this dictionary) as constructed by Shannon, was highly non-linear. To even specify this dictionary required writing down an exponential amount of information. The solution to this problem was to show that even random *linear* codes have good rate and good distance, and this at least allowed to encoding step to be performed efficiently – and gave hope that decoding could also be performed efficiently for well-structured linear codes.

2.3 Local Tree Codes

Our development of efficient coding schemes for interactive communication will follow a similar agenda as was used to obtain efficiently encodable and decodable codes in the context of a single sender and a single receiver. Our first step will be to develop a more concise representation of a good distance tree code.

Roughly, the approach is to embed a tree code into a high-girth expander. Locally, a high-girth expander is tree-like. We will associate a communication protocol with a path through this graph. And in fact, our protocol will never stray too far (when faced with a noisy channel) from the true path – the protocol will never stray so far from the path that it can traverse a full cycle and re-join the true protocol (and we would be none-the-wiser).

Definition 3. *A d-ary local tree code of girth g on alphabet Σ is a $d+1$-regular graph $G = (V, E)$ of girth at least g, and additionally each edge is labeled with a symbol from Σ*

We will define an analogous notion of the distance of a d-ary local tree code. Given a path p_1 starting at a node $u \in V$, let $W(u) \in \Sigma^{|p_1|}$ denote the set of symbols on edges traversed by p_1.

Definition 4. *A d-ary local tree code of girth g has distance α if for all nodes $u \in V$, and all simple paths p_1, p_2 of length $|p_1| = |p_2| = p < \frac{g}{2}$ starting from node u (that do not exit on the same outgoing edge), $\Delta(W(p_1), W(p_2)) \geq \alpha p$*

Note that because the graph $G = (V, E)$ has girth g, and p_1 and p_2 are simple and are restricted to have length $< \frac{g}{2}$, the paths p_1 and p_2 must be node disjoint, except at the common starting node u.

We first need to demonstrate that good d-ary local tree codes of girth $g \geq c \log n$ exist, and in this task we will face a similar obstacle to that which Schulman overcame (to show that good distance, constant sized alphabet tree codes exist). Namely, if we were to choose some high-girth, $d+1$-regular graph and we chose symbols for each edge uniformly at random from Σ, then for any node w and neighbors u and v, the edges (w, u) and (w, v) must have different symbols. Yet there is a constant failure probability, so we cannot naively use the probabilistic method.

As an alternative to Schulman's approach (based on constructing tree codes for larger and larger depth inductively), we will use the general Lovász Local Lemma to prove that good local tree codes exist. This can then be made constructive by appealing to the recent work of Moser and Tardos [10].

3 Local Tree Codes

3.1 Existence

Here we state the general version of the Lovász Local Lemma:

Let $\mathcal{A} = \{A_1, A_2, ...A_r\}$ be a finite collection of events in a probability space Ω. Then we will be interested in showing that there is some outcome in the space Ω for which none of these events occurs. The general Lovász Local Lemma gives a sufficient condition. Also let $\Gamma(A)$ be a set of events in $\mathcal{A}$ so that A is independent of $\mathcal{A} - \Gamma(A) - \{A\}$.

Lemma 1 (Lovász Local Lemma)**.** *Suppose there exists an assignment of reals $x : \mathcal{A} \rightarrow (0, 1)$ such that for all $A \in \mathcal{A}$:*

$$Pr[A] \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B))$$

Then the probability that no event in $\mathcal{A}$ occurs is non-zero:

$$Pr[\bar{A}_1 \wedge \bar{A}_2 ... \wedge \bar{A}_r] \geq \prod_{A \in \mathcal{A}} (1 - x(A))$$

Note that this probability can be exponentially small, so even if there is an outcome in Ω for which no event in $\mathcal{A}$ occurs, randomly sampling from Ω is not (necessarily) sufficient to find such an outcome with high probability.

Suppose we are given a $d+1$-regular graph $G = (V, E)$ with girth g – we will want the girth to be $\Theta(\log n)$. Our goal will be to construct a good distance local tree code on G using a constant sized alphabet. The "bad" events will be when two simple paths p_1, p_2 of the same length p that start at the same node u (and exit u using different outgoing edges) violate the distance condition - i.e. $\Delta(W(p_1), W(p_2)) < \alpha p$.

Note that the number of events is polynomially bounded. In fact, we will associate to each pair of paths p_1, p_2 as above an event A_i, and we will break the set $\mathcal{A}$ of all events up into sets $\mathcal{A}_1, \mathcal{A}_2, ...\mathcal{A}_{g/2}$ – where $\mathcal{A}_i$ is the set of all such pairs of paths that have length i. Let $|V| = n$.

Claim 1. $|\mathcal{A}_i| \leq n(d+1)d^{i-1}$

This bound is $n^{O(1)}$ for $g = \Theta(\log n)$. Note that for some event $A \in \mathcal{A}$, A is mutually independent of all other events $B \in \mathcal{A}$ in which the pair of paths in event A does not share any edge with the pair of paths in event B. Hence,

Claim 2. *$\Gamma(A)$ contains at most id^{2j} elements from $\mathcal{A}_j$ for $A \in \mathcal{A}_i$.*

We will choose $x(A) = (2d)^{-2i}$ for $A \in \mathcal{A}_i$. We first verify that this choice of $x : \mathcal{A} \to \Re$ satisfies the conditions in the Lovász Local Lemma. For an event $A \in \mathcal{A}_i$, we can compute that

$$x(A) \prod_{B \in \Gamma(A)} (1 - x(B)) \geq (2d)^{-2i} exp\{-i \sum_{j=1}^{g/2} 2^{-2j}\} \geq (2d)^{-3i}$$

And applying the Chernoff Bound for an alphabet of size $|\Sigma|$, we can upper bound the probability that A occurs for $A \in \mathcal{A}_i$ (for $\alpha = 2\eta$)

$$Pr[A] \leq exp\{-\eta^2 |\Sigma| i\}$$

So we can choose $|\Sigma| = \frac{3}{\eta^2 \log d}$, and the conditions for the general Lovász Local Lemma will hold.

Theorem 2. *For every d, n, there exist d-ary local tree codes of girth at least $c \log n$ of distance $\alpha = 2\eta$, on an alphabet of size $\frac{3}{\eta^2 \log d}$.*

3.2 Construction

In fact, we can construct local tree codes efficiently using the results of Moser and Tardos [10]:

Theorem 3 (Moser, Tardos)**.** *Suppose there exists an assignment of reals $x : \mathcal{A} \to \Re$ such that for all $A \in \mathcal{A}$:*

$$Pr[A] \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B))$$

Then there is an algorithm that computes an outcome $\omega \in \Omega$ for which on event in $\mathcal{A}$ occurs, and the expected running time of this algorithm is bounded by

$$\sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)}$$

We can apply this theorem directly to the proof in the previous section, and since $|\mathcal{A}| \leq n(d+1)^{3g}$, we obtain:

Theorem 4. *For every d, n, there is an algorithm to construct a d-ary local tree codes of girth at least $g = c \log n$ of distance $\alpha = 2\eta$, on an alphabet of size $\frac{3}{\eta^2 \log d}$ and the expected running time of this algorithm is at most $O(n(d+1)^{3g})$ which is $n^{O(1)}$ for any constant degree d.*

We can apply the results of [4] and [5] to make this construction deterministic. The running time of this deterministic construction is still polynomial in n for $g = c \log n$ and $d = O(1)$.

4 Simulation

4.1 Schulman's Protocol

Here were describe Schulman's Protocol. Let π be the noiseless channel protocol. Processor A receives input x and processor B receives input y. The history of π is described by a path (starting at the root) of a 4-ary tree $\mathcal{T}$. Each outgoing edge from an internal node (to a child) is marked by a "00", "01", "10" or "11" representing the bit sent by processor A and the bit sent by processor B in this round in the protocol.

We will let $\pi(x, \emptyset)$ denote the first bit sent by processor A, and similarly for processor B. And we let $\pi(\emptyset)$ denote the concatenation of these two bits. In this notation the second bit sent by processor A is $\pi(x, \pi(\emptyset))$, and similarly for processor B.

Schulman's Protocol (for simulating π on a noisy channel) is based on a *total information approach.* We will denote this protocol by $S(\pi)$. At each round in the simulation, each processor will maintain some pebble in the tree $\mathcal{T}$ corresponding to the noiseless protocol. Ideally, we would like the pebble for processor A and the pebble for processor B to coincide - i.e. the two processor agree on where we are in the noiseless protocol.

Since communication is subject to noise, we will expect that these two pebbles diverge. When these pebbles diverge, if we can detect this error condition, then the natural approach is to try to move these pebbles back to where the divergence occurred. Schulman's protocol is based on (often) detecting this condition, and making progress towards moving pebbles back to a "safe" location so that the simulation can continue.

For example, if the pebble for processor A, s_A, is a strict ancestor of the pebble for processor B, s_B, then we would like to move the pebble for processor B back to the pebble for processor A. If in fact neither pebble is a strict ancestor of the other, but still the pebbles disagree, we would like to move both pebbles back (closer to the root of the tree $\mathcal{T}$ for the noiseless protocol).

The tree code (and Schulman's simulation $S(\pi)$) attempts to keep track of all of the moves of each of these pebbles – not just the current location in $\mathcal{T}$, but the entire history. From the history of moves for each pebble – i.e. did the pebble move forward, backward, or hold – we can reconstruct the location of the pebble in $\mathcal{T}$. Due to communication errors in the protocol, we will sometimes not be able to recover the other processor's pebble location. Yet Schulman demonstrates that the fraction of rounds in which each processor does indeed know the location of the other processors pebble accounts for a majority of the rounds. This, as it turns out, is enough to ensure that simulation will return the correct output $\pi(x, y)$.

To this end, Schulman defines a notion of progress - which is the depth of the least common ancestor u (in $\mathcal{T}$) of s_A and s_B, minus the difference $\max(\mathit{depth}(s_A), \mathit{depth}(s_B)) - \mathit{depth}(u)$ (again each depth is computed in $\mathcal{T}$). The critical insight is that in a round in which both processors successfully determine the pebble location of the other processor, this potential function will increase. And in rounds in which this is not the case – i.e. a "bad" round – this potential function will not decrease by too much (more than an additive constant). Hence, if a large enough fraction of the rounds are "good", this potential function will increase to $|\pi|$ and the simulation $S(\pi)$ will terminate successfully.

Schulman uses the tree code to blame any bad round on some interval of errors (that contains the current round) that has too large a fraction of errors. Note that this tree code is over a different tree $\mathcal{Y}$ - which keeps track of the *total information* - i.e. what step each processor makes during each round. Naively there are six possible steps for each processor in a round - the processor can hold the pebble position, move back, or move forward in $\mathcal{T}$ on one of the four outgoing links. Additionally, we let each processor transmit the symbol that π would transmit at the location of the pebble (after either holding, moving back or moving down). So for each processor, there are twelve actions possible in each round.

4.2 Bounded Exploration

Schulman's approach is based on blaming a "bad" round on some interval of errors. We can fix the set rounds in which there is a transmission error in advance - by sampling these entries at random, and hiding this information from our protocol. Let $\vec{e}$ be the indicator vector of what rounds encounter a communication error. This will be a useful thought experiment in adapting Schulman's Protocol to work in the context of a local tree code.

Suppose a tree code has distance α. Then given a sequence of received symbols from processor B, we can compute the path in $\mathcal{Y}$ that is closest in Hamming distance to the sequence of received symbols. Let u be the end node of this path (i.e. u is a node in $\mathcal{Y}$ of depth t if we are currently in the t^{th} round of the simulation), and let v be the true node corresponding to processor B. Let w be the least common ancestor in $\mathcal{Y}$ of u and v, and let W_u, W_v be the sequence of symbols in $\mathcal{Y}$ leading from w to u and w to v respectively.

If this sequence has length r (which corresponds to an additive r difference in the depth of w compared to the depth of u and v), then at least αr transmission errors occurred in the last r transmissions – using the properties (specifically the distance) of the tree code. If $\alpha \geq 2\eta$, the chance of this occurring is exponentially small in r.

Using this observation, we can conclude that only a polynomial sized subgraph of the tree code is explored. There are only n^2 intervals, and the chance that an interval of length $c \log n$ has twice the static error rate is an inverse polynomial in n. Hence, with high probability there is no interval of length at least $c \log n$ that has at least an 2η fraction of errors.

So we could operate under the hypothesis that there is no bad interval of errors of length more than $c \log n$, and this would mean that we could never deviate more than $c \log n$ distance from the true path through $\mathcal{Y}$, if $\mathcal{Y}$ is a d-ary tree code. So there are only $d^{c \log n} = n^{O(1)}$ possibilities for the true location of processor B in $\mathcal{Y}$.

The intuition here is that if we never deviate more than $c \log n$, and we have a local tree code of girth at least $c \log n$, then this code locally looks like a tree code (because locally a high-girth graph looks like a tree, if we require paths in G to be simple).

We will formalize this argument, but for now we define the conditions on $\vec{e}$ that we require for our protocol to succeed, and a standard Chernoff Bound demonstrates that these conditions occur with high probability.

Let $g = c \log n$ be a lower bound on the girth of a local tree code.

Definition 5. *The count of $\vec{e}$ is defined as the number of intervals of length at most $\frac{g}{2}$ for which the fraction of errors exceeds 2η.*

Note that the expected count of $\vec{e}$ is approximately $O(\frac{\eta}{1-\eta} n)$ for a length n error vector $\vec{e}$.

Definition 6. *An error vector $\vec{e}$ is typical if the count is $O(\frac{\eta}{1-\eta} n)$, and no interval of length larger than $\frac{g}{2}$ has an error rate that is larger than 2η.*

Claim 3. *For $g = c \log n$ (and η sufficiently small), $\vec{e}$ is typical with high probability*

5 Local Simulation

5.1 A Protocol

Here we give a simulation protocol, that for a typical error vector, will successfully simulate the protocol π in $O(|\pi|)$ rounds of communication, for sufficiently small error rate η.

Let $G = (V, E)$ be a 13-regular graph of girth at least $g = c \log n$. Such graphs can be constructed via the probabilistic method – see Alon and Spencer [2] (or see [4]). Additionally, we designate one vertex $s \in V$ the start vertex, and for each vertex in V, we number the outgoing edges from the set $\{1, 2, ...13\}$. Edges need not be consistently number - i.e. an edge (u, v) can be number 7 for u, and 3 for v. We will call these "numbers" to distinguish from the "labels" on edges from the alphabet Σ.

Using Theorem 4, we can construct a local tree code of distance $\alpha = 4\eta$, and using an alphabet of constant size (for any fixed, but sufficiently small error rate η). We will describe the local simulation protocol from the perspective of processor A, but an analogous protocol will hold for processor B.

Processor A will maintain two state markers t_A and t_B in the graph G, and a set of live paths (representing alternatives for the total path through G taken by processor B). Let $\mathcal{P}$ be the set of live paths. Additionally, at the end of each round, processor A will move the pebble (maintained in $\mathcal{T}$, the tree corresponding to the noiseless protocol) and will transmit a bit (corresponding to the bit that would be transmitted by processor A during the noiseless protocol, specified by the node in $\mathcal{T}$ at which processor A's pebble resides). This output corresponds to a choice of a number in the set $\{1, 2, ...12\}$. We will use this choice to move the state marker for processor A, but since we require the path we traverse in G to be simple, one of the thirteen outgoing edges cannot be traversed. So deleting this number from the set $\{1, 2, ...13\}$, if processor A's action corresponds to $i \in \{1, 2, ...12\}$, we will choose the i^{th} smallest number in the set $\{1, 2, ..., 13\}$ excluding the number of the edge that cannot be traversed. We will use this number of choose the corresponding outgoing edge for the node at which marker t_A is currently placed. For ease of exposition, we will call this move the numbered edge corresponding to processor A's action. Note that this depends on which edge was traversed by t_A in the previous round.

1. Initialize $\mathcal{P}$ to be the single path (no edges) starting at node s, place both state markers t_A and t_B at s

2. Given the just received symbol from processor B, augment each path in $\mathcal{P}$ by each possible outgoing edge that could be traversed in the next move (such that the path is still simple)

3. **Trim:** Delete each path in $\mathcal{P}$ for which there is an interval of length larger than $\frac{g}{2}$ which has an error rate (compared to the received symbols during that interval) larger than 2η

4. **Decode:** Choose the path $p \in \mathcal{P}$ that is closest in Hamming distance to the received set of symbols (so far). This path encodes (a hypothesis for) the location of marker t_B

5. Let s_A, s_B be the pebble location (in $\mathcal{T}$) for processor A and for processor B (based on the hypothesis t_B).

   If s_A is a strict ancestor of s_B, hold the pebble location

   If s_B is a strict ancestor of s_A, move the pebble back

   If $s_A = s_B$, decode the just received symbol from processor B, and move down $\mathcal{T}$ according to the bit sent in the previous round by processor A and the hypothesis for the received bit from processor B

6. Traverse the numbered edge corresponding to processor A's action, and transmit the symbol on this edge

5.2 Analysis

Here we prove that the local protocol given in the previous subsection does simulate π correctly for a typical error vector $\vec{e}$, and furthermore that this local simulation protocol $L(\pi)$ runs in time polynomial in $|\pi|$.

Definition 7. *We will call a round i "bad" if the state location s_B decoded in this round is not correct*

We will bound the number of "bad" rounds by the count of $\vec{e}$, provided that $\vec{e}$ does not contain any subsequence of length larger than $\frac{g}{2}$ with an error rate larger than $2\eta = \frac{\alpha}{2}$. In order to accomplish this goal, we will first analyze properties of the set of live paths $\mathcal{P}$. Let p_B be the true path of the state marker for processor B, and let $p \in \mathcal{P}$ be any live path in the course of the simulation $L(\pi)$.

Lemma 2. *If $\vec{e}$ is typical, either $p = p_B$ or p agrees with p_B until the last r moves and during the last r moves, p is node disjoint from p_B for $r < \frac{g}{2}$*

Proof: Note that both p and p_B are simple walks, in a girth g graph. So if p first disagrees with p_B at round j, then p cannot agree with p_B until at least round $j + \frac{g}{2}$. Let p_1 and p_2 be the subpaths of p and p_B respectively, starting at round j and continuing until round $j + \frac{g}{2}$. Using the properties of a local tree code, the fractional Hamming distance between the sequence of symbols on these paths is at least α, so if the received set of symbols during the interval from j to $j + \frac{g}{2}$ does not result in deleting p from $\mathcal{P}$, then the fraction of errors during this interval is larger than 2η, and hence $\vec{e}$ is not typical because this interval is length $\frac{g}{2}$. Hence p can only be currently live (at round i) if p agrees with p_B until the last r moves and during the last r moves for some $r < \frac{g}{2}$, and so p and p_B must be node disjoint on these last r moves because G has girth at least g. $\square$

Corollary 1. *At any point in the local simulation, $|\mathcal{P}| \leq (d+1)^{\frac{g}{2}}$, and this is $n^{O(1)}$ for $g = c \log n$ and d is constant.*

Lemma 3. *The number of "bad" rounds is bounded by the count of $\vec{e}$, if $\vec{e}$ does not contain any subsequence of length larger than $\frac{g}{2}$ with an error rate larger than 2η.*

Proof: Suppose, in some round j, the path "decoded" p is different from the true path p_B. Using Lemma 2, p agrees with p_B until the last r moves and during the last r moves, p is node disjoint from p_B for $r < \frac{g}{2}$. Hence, the last r symbols received must be closer in Hamming distance to the sequence corresponding to the last r moves in p (call this path p_1) than to the last r moves in p_B (call this path p_2). This implies that there are more than a 2η fraction of errors in this length r interval. Hence we can "blame" this decoding error on this interval of rounds $j - r + 1$ to j. Note that this interval is never "blamed" more than once, because an interval can only be "blamed" by a "bad" round corresponding to the end of the interval. $\square$

Analogously to Schulman's protocol, we can define a potential function, which we call the *rank*, as the the depth of the least common ancestor u (in $\mathcal{T}$) of s_A and s_B, minus the difference $\max(depth(s_A), depth(s_B)) - depth(u)$ (again each depth is computed in $\mathcal{T}$). Note that in this potential function, s_A and t_A represent the true locations of the pebble for processor A and the pebble for processor B respectively.

Claim 4. *In a "good" round, the rank increases by at least one and in a "bad" round the rank can increase by at most three.*

Finally, note that the least common ancestor of s_A and s_B (where these are the true pebble locations) must always be on the path of $\pi(x, y)$ in $\mathcal{T}$, because a processor moves its own pebble according to the bit that it attempted to send, so both pebbles cannot exit the path of $\pi(x, y)$ on the same edge in $\mathcal{T}$. So this implies that when the rank reaches $|\pi|$, the local simulation $L(\pi)$ has computed the correct output $\pi(x, y)$.

Using Lemma 3, and the definition of a typical error vector $\vec{e}$, it follows that the rank of the local simulation (at termination) is at least $|\pi|$, and this proves correctness. And we can also apply Corollary 1, which implies that the protocol $L(\pi)$ can be implemented in time polynomial in $|\pi|$. This implies our main theorem.

6 Open Questions

Open Question 1. *Can local tree codes be used to give efficient coding (for interactive communication) for worst-case channels?*

The crux of this question is that if an adversary is allowed to distribute errors in bursts, then the naive searching procedure for finding the correct path will run in exponential time. Local tree coding can still be a useful construct in this more general setting if decoding the correct path can be implemented more cleverly (perhaps using a nice, explicit construction of a high-girth expander as opposed to a randomly chosen one). Since local tree codes have good distance when comparing paths that do not share internal nodes, we can extend this distance bound to paths that do cross and so local tree codes still remain an exponentially more concise way to represent the necessary properties of a tree code – however, now only decoding (in the worst-case channel model) is the computational bottleneck.

In fact, an easier open question is:

Open Question 2. *Can we efficiently code for interactive communication in noisy, but more general models than the binary symmetric channel?*

For example, answering this question in the context of bursty channels or in the context of channels represented by a Markov chain may already require interesting new ideas for decoding. Another intriguing open question is whether local tree codes can be used to reach the channel capacity in an interactive setting. The core of the argument is based on high-girth graphs, and in the context of error correcting codes, the girth of the Tanner graph plays a fundamental role in efficiently reaching the coding capacity of a channel. So we could ask if girth plays a similar role in the context of coding for interactive communication.

Acknowledgements

I would like to thank Mark Braverman for introducing me to these questions, and Boaz Barak for helpful discussions.

References

[1] N. Alon, S. Hoory, and N. Linial. The moore bound for irregular graphs. *Graphs Combin*, pages 53–57, 2002.

[2] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley Interscience Series, 2000.

[3] M. Braverman and A. Rao. Towards coding for maximum errors in interactive communication. *STOC*, 2011, to appear.

[4] S. Chandran. High girth graph construction. *SIAM Journal on Discrete Math*, pages 366–370, 2003.

[5] K. Chandrasekaran, N. Goyal, and B. Haeupler. Deterministic algorithms for the lovasz local lemma. *SODA*, pages 992–1004, 2010.

[6] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley Interscience Series, 1991.

[7] P. Erdos and L. Lovasz. Problems and results on 3-chromatic hypergraphs and some related question. *Infinite and Finite Sets*, pages 609–627, 1975.

[8] R. Gallager. Finding parity in a simple broadcast network. *IEEE Transactions on Information Theory*, pages 176–180, 1988.

[9] R. Gelles and A. Sahai. Potent tree codes and their applications: Coding for interactive communication, revisited. *http://arxiv.org/abs/1104.0739*, 2011.

[10] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[11] R. Moser and G. Tardos. A constructive proof of the general lovasz local lemma. *JACM*, pages 1–15, 2010.

[12] L. Schulman. Communication on noisy channels: a coding theorem for computation. *FOCS*, pages 724–733, 1992.

[13] L. Schulman. Deterministic coding for interactive communication. *STOC*, pages 747–756, 1993.

[14] L. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, pages 1745–1756, 1996.

[15] C. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, pages 623–656, 1948.

[16] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, pages 533–547, 1981.

[17] J. Wozencraft. Sequential decoding for reliable communications. *MIT Technical Report*, 1957.

[18] A. Yao. Some complexity questions related to distributive computing. *STOC*, pages 209–213, 1979.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de