

Coding for Interactive Communication

Leonard Schulman

EE150
Vijay Gupta

The Problem

- Let the input to a computation problem be split between two processors connected by a communication link.
- Let an interactive protocol π be known by which, on any input, the processors can solve the problem using no more than T transmissions of bits between them, if the channel is noiseless in either direction.
- If the channel is noisy, what is the effect upon the number of transmissions needed in order to solve the computation problem reliably?

Approaches That Spring to Mind

- Naive approach: Send every message many times and take a majority vote.
 - “Rate” of simulation (number of protocol steps simulated per transmission) goes to zero as T increases and even if only a constant probability of error is desired.
- Shannon’s approach: Instead of treating every bit separately, jointly encode large blocks of data into long codewords. The error probability goes down exponentially in T while the time required for transmission compared to the T transmissions in the noiseless case decreases only by a constant factor (capacity of the channel).

Is Shannon's Solution Relevant?

- In new applications of communication (distributed computing, VLSI chips, control), Shannon's work is a necessary but not sufficient condition for sustained interaction and computation. There are additional complications like
 - Processors do not know what they want to transmit more than one bit ahead, so block codes cannot be used.
 - Once an error has occurred, subsequent exchanges are affected. Yet the processors ought to be able to recover following any sequence of errors.

Notation and Assumptions: Part I

- Assume unless stated otherwise that processors are connected by a pair of unidirectional channels which, in every unit of time, transmit one bit in each direction.
- Call the number of bit exchanges required for a protocol to terminate on any input as its run time.
- Notation: Let $g : N \rightarrow N$ be a non-decreasing function. Then
 - $\Omega(g)$ is the set of functions $f : N \rightarrow N$ for which $\lim_{i \rightarrow \infty} \inf f(i)/g(i) > 0$.
 - $O(g)$ is the set of functions $f : N \rightarrow N$ for which $\lim_{i \rightarrow \infty} \sup f(i)/g(i) < \infty$.
 - $\theta(g) = O(g) \cap \Omega(g)$ denotes the set of functions which grow proportionately to g .

Main Results: Theorem 2 of the Paper

- Theorem 2 says that if in each direction between a pair of processors a binary symmetric channel of capacity C be given, then there is a deterministic communication protocol which given any noiseless channel protocol π running in time T , simulates π on the noisy channel in time $\theta(T/C)$ with error probability $e^{-\Omega(T)}$.
- Note that if π is deterministic, so is the simulation; if π needs some randomness, exactly the same randomness resource is required for the simulation.
- The probability of error of the simulation is bounded by the sum of the probability of error of π and the probability of error of the simulation.
- Analog of Shannon's coding theorem.

Main Results: Theorem 3 of the Paper

- Theorem 3 says that given an oracle for a tree code, the expected computation time of each of the processors implementing the protocol, when the communication channels are binary symmetric, is polynomial in T .
 - Tree codes will be described later.
 - Oracle is a procedure which on request, produces a local description of the tree code while charging one unit of time for this.
 - Thus any polynomial time algorithm for this task will result in polynomial time encoding and decoding procedures.
- This approach basically says that every upper bound achieved in the noiseless case leads to an upper bound in the noisy channel case as well.

Main Results: Theorem 4 of the Paper

- Theorem 4 pertains to adversarial channels.
- It says that there is a deterministic communication protocol which given any noiseless channel protocol π running in time T , runs in time $N = \theta(T)$ and successfully simulates π provided the number of incorrectly transmitted bits is at most $N/240$.
- Note that due to the limit on the tolerable noise level, this result does not dominate Theorem 2.

The Model

- Function $f(z)$ needs to be computed by two processors A and B .
- The argument $z = (z_A, z_B)$ is split between the two processors and there is a link between them.
- There is no time or space resource constraint.
- Randomness is a resource with three possible scenarios:
 - No processor needs any randomness.
 - Each processor has an unbounded source of random coins but cannot see the other processor's coins.
 - There is a common unbounded source of random coins.

The Channel

- The binary symmetric channel is a synchronous communication link which in every unit of time accepts either a 0 or a 1 and outputs either a 0 or a 1.
- With probability ϵ , the output differs from the input; else it is the same.
- The errors are independent of everything else; in particular of other error events (memoryless channel).
- Average mutual information between two ensembles $\{P(x)\}_{x \in X}$ and $\{P(y)\}_{y \in Y}$ with a joint distribution $\{P(xy)\}_{x \in X, y \in Y}$ is defined as $\sum_{xy} P(xy) \log[P(xy)/P(x)P(y)]$.
- Intuitively it is a measure of how much information is provided about one ensemble by the specification of a randomly chosen point of the other ensemble.

Capacity of the Channel

- Suppose we wish to find the maximum amount of information transmitted from input to output.
- We try to find a source just suitable for the channel by maximizing the average mutual information between the input and output distributions over all probability distributions of the inputs.
- In the case of the binary symmetric channel the capacity (in base 2) is $C = \epsilon \log(2\epsilon) + (1 - \epsilon) \log(2(1 - \epsilon))$.
- A noiseless channel has $C = 1$ and in general, $0 \leq C \leq 1$.

Tree Codes

- Let S be a finite alphabet.
- If $s = (s_1, s_2, \dots, s_m)$ and $r = (r_1, r_2, \dots, r_m)$ are words of the same length over S , define the (Hamming) distance $\Delta(s, r)$ between s and r as the number of positions i in which the words differ.
- A d -ary tree of depth n is a rooted tree in which every internal node has d children, and every leaf is at depth n (counting from 0 for the root).
- A tree code has four parameters - $\{d, S, \alpha, n\}$.
- A d -ary tree code over alphabet S of distance α and depth n is a d -ary tree of depth n in which every arc of the tree is labeled with a character from the alphabet S subject to the following condition.

The Condition

- Let v_1 and v_2 be any two nodes at some common depth h in the tree.
- Let $h - l$ be the depth of their least common ancestor.
- Let $W(v_1)$ and $W(v_2)$ be the concatenation of the letters on the arcs leading from the root to v_1 and v_2 respectively.
- The Condition: The distance between $W(v_1)$ and $W(v_2)$ should be at least αl . Note that a truncation of a tree code is also a tree code.
- From now on we fix α to be $1/2$.

A Key Property of Tree Codes

- The key property of tree codes, is that the alphabet size required to guarantee their existence does not depend on n . In fact, for any fixed d there exists an infinite tree code with a constant size alphabet.
- Lemma 1 says that for any fixed degree d , there exists a finite alphabet which suffices to label the arcs of a d -ary tree code. Specifically,
 - An alphabet S of size 99 suffices to label the arcs of a binary tree code of distance parameter $1/2$ and any depth n .
 - Let $\eta(\alpha) = 1/(1 - \alpha)^{1-\alpha}(1/\alpha^\alpha) \leq 2$. Then an alphabet S of size $2\lfloor 2\eta(\alpha)d^{1/(1-\alpha)} \rfloor - 1$ suffices to label the arcs of an infinite d -ary tree code of distance parameter α .
- Proof of the first part follows by induction on n .

Channel Capacity

- Lemma 2 says that let a binary symmetric channel M of capacity C and an alphabet S be given. Then there is a transmission code for the alphabet, ie, a pair consisting of an encoding function taking S to n -tuples of 0's and 1's, and a corresponding decoding function such that $n = O((1/C)\log|S|)$, while the probability of error in any transmission over the channel is no more than $2^{-208}3^{-40}$.
- Role of channel capacity should not be over-emphasized in this situation.

The Noiseless Case, aka, Notations: Part II

- Let π be the noiseless channel protocol of length T to be simulated.
- Take the hardest case, in which every transmission of π is only one bit long. Assume that in every round both processors send a bit.
- The history of π on any particular input is described by a path from the root to a leaf, in a 4-ary tree T , in which every arc is labeled by one of the pairs 00, 01, 10 and 11 referring to the pair of messages that can be exchanged by the processors in that round.
 - Let $x = x_A x_B$ be the problem input.
 - Then let $\pi_A(x_A, \phi)$ denote the first bit sent by processor A .
 - Let $\pi_B(x_B, \phi)$ denote the first bit sent by processor B .

- Let let $\pi(x, \phi)$ be the pair of these first-round bits.
- Since there is no noise, in the second round of π , both processors are aware of the bits exchanged in the first round.
- In general, if messages m_1, \dots, m_t (each $m_i \in \{00, 01, 10, 11\}$) have been exchanged in the first t rounds, then the transmissions in the round $t + 1$ are denoted by $\pi_A(x_A, m_1, \dots, m_t)$ and $\pi_B(x_B, m_1, \dots, m_t)$ and the pair of bits by $\pi(x, m_1, \dots, m_t)$.
- The vertex of the tree T labeled by a sequence of exchanges m_1, \dots, m_t is denoted by $T[m_1 : \dots : m_t]$. For example, the root is $T[]$.
- The protocol π specifies a path from the root to a leaf in T on being specified an input. We call this path γ_x .
- At the end, an output is determined.

Noiseless v/s Noisy Case

- On a noiseless channel, the processors extend γ_x by one arc in every round.
- In the noisy channel simulation, the processors will occasionally go along offshoots from the true path.
- They must develop γ_x without expending too much time pursuing other branches of T .

The Simulation Protocol

- The simulation protocol attempts to recreate the development by π of the path γ_x in T on being given the input x .
- Equip each processor with a pebble which it moves about on T , in accordance with its best guess to date regarding γ_x .
- In every round, each processor sends the bit it would transmit in π upon reaching the current pebble position in T among other information.
- When the pebbles coincide, this exchange is a useful simulation of a step in π , unless the exchange fails.
- Due to channel errors, the pebbles would tend to diverge.
- The simulation embeds π within a larger protocol that provides a restoring force to negate the divergence.

Mechanics of the Protocol

- In any round of the protocol, a processor can move its pebble only to a neighbor of the current position in T or leave it unmoved.
- So there are six possibilities $\{00, 01, 10, 11, H, B\}$. H refers to 'hold', B to 'back' toward the root.
- Following such a move, either a 0 or a 1 is transmitted according to the value transmitted in π at that point.
- The entire history of any one processor up through time t can therefore be described as a sequence of t tracks each an integer 1-12 indicating the pebble move made by the processor in some round, and the value at the vertex reached by the pebble after that move.
- Denote the sequence of tracks taken by a processor as its state.

Mechanics of the Protocol

- The possible states for a processor is described by a 12-ary tree Y . In each round, each processor moves to a child of its previous state in this state tree.
- For each state $s \in Y$ call $\text{pebble}(s) \in T$ as the pebble position of a processor which reaches state s .
- We want the processors to keep each other informed of their exact state, and thus of their entire history. Since only a constant slowdown is desired, we cannot simply encode each track with a transmission code and send it since the occasional errors would affect the entire future.

State Tree, aka, Notations and Definitions: III

- Following Lemma 1, we can identify with Y a 12-ary tree code of depth $5T$ over a suitable alphabet S .
- If τ_1, \dots, τ_t be a sequence of t tracks (indicating a processor's actions in the first t rounds), call
 - the vertex of Y reached from the root by the sequence of arcs τ_1, \dots, τ_t by $Y[\tau_1 : \dots : \tau_t]$.
 - The letter on the arc $Y[\tau_1 : \dots : \tau_{t-1}]Y[\tau_1 : \dots : \tau_t]$ of the tree code by $w(\tau_1 \dots \tau_t) \in S$.
 - The concatenation $w(\tau_1)w(\tau_1\tau_2) \dots w(\tau_1 \dots \tau_t)$ by $W(\tau_1 \dots \tau_t) \in S^t$.

The Prescription

- The prescription for transmitting messages in the protocol is this.
 - Suppose processor A in round t decides on a pebble move $\alpha \in \{00, 01, 10, 11, H, B\}$ and reaches vertex v of T with that move.
 - Thus its track is $\tau_t = \alpha \times \pi_A(x_A, v)$ corresponding to an integer between 1 and 12.
 - Let its past tracks be $\tau_1, \dots, \tau_{t-1}$.
 - Then it transmits in round t , the letter $w(\tau_1 \dots \tau_t)$.

Lemma 3

- Let $\tau_1 \dots \tau_t$ be a sequence of tracks (a state) and $Z = Z_1 \dots Z_t \in S^t$ be a sequence of letters. Denote
 - The probability that Z is received over the channel given that the sequence $W(\tau_1 \dots \tau_t)$ was transmitted by $P(Z|W(\tau_1 \dots \tau_t))$.
 - The probability that $Z_r \dots Z_t$ are the last $t - r + 1$ characters received in a transmission of $W(\tau_1 \dots \tau_t)$ by $P(Z_r \dots Z_t|W(\tau_1 \dots \tau_t))$.
- Since the channel is memoryless, Lemma 3 says that $P(Z|W(\tau_1 \dots \tau_t))$ is bounded above by $(2^{-208} 3^{-40})^{\Delta(W(\tau_1 \dots \tau_t), Z)}$. More generally,
$$P(Z_r \dots Z_t|W(\tau_1 \dots \tau_t)) \leq (2^{-208} 3^{-40})^{\Delta(w(\tau_1 \dots \tau_r) \dots w(\tau_1 \dots \tau_t), Z_r \dots Z_t)}.$$

Pebble Moves

- A and B act symmetrically. Let $Z \in S^{t-1}$ be the sequence of letters received by A up to and including round $t - 1$.
- A determines which state $\tau_1 \dots \tau_{t-1}$ of processor B at depth $t - 1$ would minimize the distance $\Delta(W(\tau_1 \dots \tau_{t-1}), Z)$ and chooses that state g as its best guess for B 's current state. A correct guess implies correct determination of B 's pebble position.
- Suppose A 's pebble is at vertex $v \in T$. Thus it has just sent the bit $\pi_A(x_A, v)$ to B .
- Let b be the corresponding bit which A has decoded as B 's proposed message in π .
- Also let $g \in Y$ be A 's best guess about B ' state.

- A compares the positions of v and $\text{pebble}(g)$ in T .
 - If they are the same, A interprets b as the bit B would transmit at v in protocol π . Thus it moves its pebble down in T on the arc labeled by the pair of bits given by b and its own choice $\pi_A(x_A, v)$.
 - On the other hand, if they are different, A tries to close the gap by making the pebbles meet at their least common ancestor. Two cases arise
 - * v is not an ancestor of $\text{pebble}(g)$. Then A moves its pebble toward the root in T (Back).
 - * v is an ancestor. A cannot do anything. It stays put and hopes B will bring its pebble back to v . (Hold)

Why the Least Common Ancestor?

- Lemma 4 says that the least common ancestor of the two pebbles lies on γ_x which is the path that the noiseless protocol followed on T .
- Proof is basically that the pebble move rules ensure that the arcs leading toward a pebble are always correctly labeled in the bit corresponding to that processor's actions in π .
- A vertex z of T is on γ_x iff for every strict ancestor z' of z , the arc leading from z' toward z is that specified by both processors' actions in π at z' , which is nothing but the arc labeled by the pair of bits $\pi_A(x_A, z')$ and $\pi_B(x_B, z')$.

Summary of the Simulation Protocol

- Assume the perspective of A .
- Let T be the length of the protocol π . Then repeat the following $N=5T$ times.
- Begin with its own state s_A at $Y[H \times \pi_A(x_A, \phi)]$ and its own pebble at the root of T .
 - Transmit $w(s_A)$ to processor B . Remember $w(\tau_1 \dots \tau_t)$ was the letter on arc $Y[\tau_1 : \dots : \tau_{t-1}]Y[\tau_1 : \dots : \tau_t]$ of the tree code.
 - Given the sequence of messages Z received to date from processor B , guess the current state g of B as that minimizing $\Delta(W(g), Z)$.
 - Compute from g both $\text{pebble}(g)$ and the bit b representing a message of B in π .

- Depending on the relation of v to $\text{pebble}(g)$, do one of the following
 - * If v is a strict ancestor of $\text{pebble}(g)$, pebble move is H . Next track is $\tau = H \times \pi_A(x_A, v)$. Reset the state s_A to $s_A\tau$.
 - * If least common ancestor of v and $\text{pebble}(g)$ is a strict ancestor of v , pebble move is B . Next track is $\tau = B \times \pi_A(x_A, v : B)$. Reset s_A to $s_A\tau$.
 - * If $v = \text{pebble}(g)$ then pebble move is according to the pair of bits $(\pi_A(x_A, v), b)$. Next track is $\tau = (\pi_A(x_A, v), b) \times \pi_A(x_A, v : (\pi_A(x_A, v), b))$. Reset s_A to $s_A\tau$.
- If π completes before $5T$, processors keep sending 0's.

Analysis

- Proposition 1 says that the simulation protocol when run for $5T$ rounds on any noiseless channel protocol π of length T , will correctly simulate π except for an error probability no greater than 2^{-5T} .
- – Let v_a and v_b be the positions of the two pebbles in T after some round of simulation. with the least common ancestor at \bar{v} .
 - The current mark of the protocol is the depth of \bar{v} minus the distance from \bar{v} to the further of v_a and v_b .
 - Thus if the mark at termination is at least T , then π is successfully simulated.
 - A round is good if both processors guess the other's state correctly. Else the round is bad.

Analysis

- Lemma 5 says that the pebble moves in a good round increase the mark by 1, while in a bad round decrease the mark by at most 3.
- Proof is obvious since
 - After a good round, processors are using correct information regarding both pebble positions and bits transmitted in π .
 - * If pebbles were located at same vertex, they move to the same child.
 - * If not, then the pebble not equal to \bar{v} moves one arc closer to it.
 - In a bad round, each pebble, and hence \bar{v} moves by at most one arc.

Analysis

- Since the simulation is run for $5T$ rounds, corollary 1 says that the simulation is successful if the fraction of bad rounds is at most $1/5$.
- How can we bound the number of bad rounds?
- For any round t , for both processors measure the magnitude of the error regarding the other's state.
- This is essentially the difference between t and the level of the least common ancestor of the true state and the guessed state.
- Call $l(t)$ as the greater of these magnitudes.
- A round is good if $l(t) = 0$.

Analysis

- Now a wrong guess of magnitude l must be at a Hamming distance of at least $l/2$ from the correct guess, from the tree code condition.
- Also in order to be preferred to the correct guess, at least $l/4$ character decoding errors are required within the last l rounds.
- Lemma 3 immediately leads to Corollary 2, which says that conditioned on any pair of states the processors may be in at time $t - l$, the probability of a processor making a particular wrong guess of magnitude l at time t is at most $2^l (2^{-208} 3^{-40})^{l/4} = (2^{-51} 3^{-10})^l$.

Analysis

- Now define the error interval corresponding to a bad round at time t as the sequence of $l(t)$ rounds prior to and including t .
- Every bad round is contained inside an error interval, hence the size of the union of the error intervals bounds from above the number of bad rounds.
- Lemma 6 says that if a set of erroneous guesses defines a disjoint set of error intervals, of lengths l_1, \dots, l_k , then the probability that these erroneous guesses occur is at most $(2^{-50}3^{-10})^{\sum l_i}$.

Analysis

- Proof is by induction on the number of error intervals.
- If the last error interval is due to a wrong guess at round t of magnitude l_k and s and r be the true state of other processor at time t and its erroneous guess, then the transmitted strings $W(s)$ and $W(r)$ differ only in the last l_k rounds.
- Corollary 2 can then be used to bound the probability of occurrence of erroneous guesses.

Analysis

- Lemma 7 says that in any finite set of intervals on the real line whose union J is of total length s there is a subset of disjoint intervals whose union is of total length at least $s/2$.
- The proof is basically showing that J can be written as the union of two sequences of disjoint intervals.
- Only interesting case is the one in which the intervals of the family are closed and their union J is an interval.
 - First put into the first sequence that interval reaching the left endpoint of J and extending furthest to the right.
 - In successive steps, select the interval intersecting the union of those selected so far, and extending furthest to the right; adjoin the new interval to one of the sequences in alternation.

Analysis

- Lemma 7 and Corollary 1 together imply that it is sufficient to bound the probability of occurrence of some set of erroneous guesses which define disjoint error intervals spanning at least $N/10$ rounds, $N=5T$.
- There are at most 2^{2N} ways in which disjoint error intervals can be distributed in the N rounds available.
- For each error of magnitude l the erroneous guess can be at one of $12^l - 1$ states.
- So for a given set of disjoint error intervals there are at most 12^N ways in which the guesses can arise.
- For a set of erroneous guesses that defines a set of disjoint error intervals of lengths l_1, \dots, l_k the probability of all these guesses occurring is at most $96^{-10 \sum l_i}$.

- If these error intervals span at least $N/10$ rounds, the probability is at most 96^{-N} .
- Ranging over all such events, we find that the probability of the protocol failing to simulate π is at most $2^{-N} = 2^{-5T}$.
- This proves Proposition 1.

Adversarial Channels

- For proving Theorem 4, consider that for the transmission code used to exchange individual tracks between the processors, we use a code with a minimum distance property.
- So that any two codewords differ in at least $1/3$ of their length.
- Now for one of the processors to make a wrong guess of magnitude l , at least $l/4$ of the transmitted characters must have been received incorrectly. Thus the fraction of incorrectly transmitted bits during this period of length l must be at least $1/24$.
- We proceed as in proposition 1, and upper bound the total length of disjoint error intervals by $N/10$.
- For this it suffices that the fraction of bit errors during the protocol be bounded by $1/240$.

Sections 4 and 5

- Section 4 deals with the computational overhead of the protocol and proves theorem 3.
- Section 5 proves results for Binary Erasure Channel. Although elegant results are obtained, the methods involved are not useful in more practical error models.

Discussion

- Essentially a coding theorem for computation has been presented.
 - Like Shannon's framework, the proof is not constructive.
 - In Shannon's work any code random enough could do the job, but here the problem is tougher.
- Explicit construction of a tree code means that each label be computable in time polynomial in the depth of the label.
 - Explicit constructions are known with alphabet-size polynomial in the depth of the tree.
 - The existence proof can be adapted into a construction if we only require Hamming distance condition to hold for pairs of vertices whose distance from their least common ancestor is at most logarithmic in the depth of the tree.

Discussion

- Lemma 2 involves the constant $2^{-208}3^{-40}$. This probability is too low for any practical rates to hold. Some optimization of the parameters might be required.
- – For data transmission, it is possible to communicate as close to capacity as desired.
- On a BSC, it has been shown that interactive protocols can be deterministically simulated at a rate within a constant factor of channel capacity.
- Does the gap always hold or can a better simulation close it?

Discussion

- Shannon's theorem has a converse which says that at any transmission rate beyond channel capacity, a code must suffer high error probabilities on at least some codewords.
- But in our case, there might be problems which can be simulated with relatively greater efficiency in the noisy case.
- For instance, a new protocol can be devised for the case of noisy channels that is not a simulation of a noiseless channel protocol and that takes advantage of the greater number of rounds available.
- Thus randomness is a resource which must be handled carefully. If only deterministic complexities are compared, one should be careful of processors 'gaining' random bits from the noise on the channel.

Discussion

- Another complicating factor to study for various channels is that for a channel with memory, capacity can be increased with feedback.
- Hence there may be a trade-off between the use of channel as a transmission medium and as a means of amplifying the capacity of the channel in the opposite direction.