# Comparative Discussion of Coding Scheme in Relation to Braverman and Rao's Algorithm

Tajbeed Ahmed Chowdhury
*2576653*
*Saarland University*
Saarbrücken, Germany
s8tachow@stud.uni-saarland.de

Koushik Chowdhury
*2572865*
*Saarland University*
Saarbrücken, Germany
s8kochow@stud.uni-saarland.de

*Abstract*— **This report discusses Braverman and Rao's coding scheme [1], which has optimal noise resilience while maintaining a constant rate. They demonstrated that any communication protocol between two parties can be encoded in such a way that the protocol is successful even if a fraction of all symbols transmitted by the parties is corrupted adversarially, ¼ - $\epsilon$ to be exact. This report also provides a quick overview of Shannon coding, interactive coding, and tree codes to help the reader understand why such a protocol is necessary and to explain it as simply as possible.**

*Index Terms*— **Shannon, Interactive, Coding, Protocol, Tree, Transmission.**

## I. CLASSICAL COMMUNICATION

### A. A Classical Code Problem

Consider the following scenario. Suppose there are two people named Allen and Bella where Allen is the sender and Bella is the receiver. Allen wants to send Bella a binary message. Allen sends Bella a message with the number 101010, but Bella receives a message with the number 10100, which is not the original message. This indicates that Bella received the wrong message. The question now is how to properly encode the message so that an error can be recovered. This can be solved by Shannon's 2nd theorem.

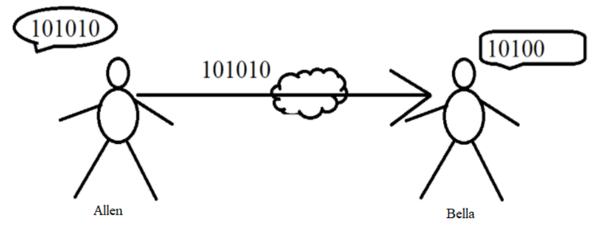

Fig. 1. A Classical Problem

### B. Shannon's 2nd theorem

This is the problem of coding on a noisy channel. The goal is to protect information from transmission errors by encoding it. We are looking at a source that has been optimally or near optimally coded [2]. In the following figure, we can see an encoder and a decoder, with the encoder sending a message from the source string to the decoder over a noisy channel. Consider C as the channel capacity in terms of channel probabilities. Let's say the source string sends 'b' bits messages to the encoder, which converts the messages into a codeword of b/R bits, where R is the information rate, and for some rate $0 < R < 1$. By using a channel, an

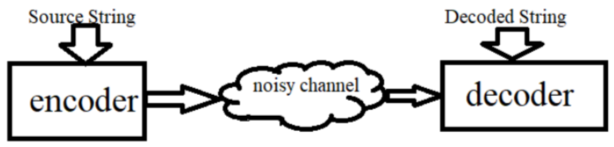

Fig. 2. Messages from encoder to decoder

information rate, R can be achievable, if there is an encoding such that as b increases, the probability of error decreases [3]. According to the theorem, if the information rate, R, is less than the channel capacity, C, the information rate, R, is achievable, and if the information rate, R, is greater than the channel capacity, C, the information rate, R, is not achievable. In other words, If there is a sequence of codes for increasing code-word lengths n, such that the maximum error goes to zero as n goes to infinity, an information rate, R can be achieved [3]. This is also known as the Channel coding theorem.

## II. INTERACTIVE COMMUNICATION

What if, when sending a message to a recipient, the sender also wants to know whether the recipient received it or not? We can say that we're interested in seeing how the sender and receiver interact with each other. For that, we'll need interactive communication. Consider the problem of a chess board [4]. Assume Allen and Bella want to play chess together but are not in the same location, so they use "telephone communication" as their medium. Allen moves his knight to E3 to F5 but Bella hears E3 to G2. Allen checkmates Bella a few moments later, and Allen wins the game. Allen tells Bella that the game is over and that he has won the game, but Bella notices that it is still going on and that she has a chance to escape and not lose the game. That means Allen and Bella's chess boards have different scenarios. This occurs because Allen and Bella

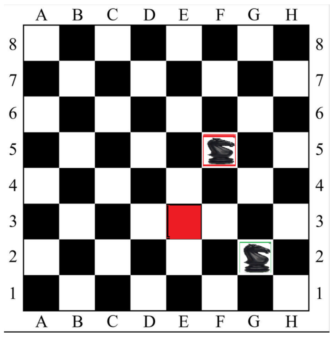

Fig. 3. Chess Board problem

communicate over a noisy channel, and there were some phone line interruptions when Allen instructed Bella to move her knight E3 to F5 and Bella misheard. The simple rule is that both Allen and Bella must communicate the information to each other, but this did not occur due to a wrong move. So, what is the best way to solve this problem so that both parties can communicate without being interrupted by errors? Interactive coding is the answer. Interactive coding is a modified version of classical code. Both parties can communicate with each other in the presence of noise using interactive coding. Let's take a look at interactive coding. Consider the following scenario. Assume that there is an input to a computation problem that is split between two processors and connected by a communication link. Because the processors have an interactive protocol and the channel is noiseless, they can solve the computation problem with no more than T bit transmissions between them [5]. How many transmissions would be required to solve this computation problem if the channel was noisy? When you think about this problem, there are basically two traditional approaches that come to mind. Shannon's approach and Naive approach. In the Naive Approach, we repeatedly send the same messages, causing rate R to zero and the number of transmissions to increase. However, in the Shannon Approach, large blocks of data are jointly encoded into long codewords, lowering the probability of error [5]. The problem is that for applications such as VLSI chips and distributed computing, the Shannon Approach is insufficient to ensure uninterrupted interaction and computation. The reason for this is that large blocks of data cannot be used because processors cannot transmit more than one bit ahead of time, and if an error occurs, upcoming reciprocity may be harmed. Consider the chess problem once more [4]. There is interactive communication that is carried out via protocol, $\pi$. $\pi$A stands for Allen, $\pi$B for Bella, and Allen and Bella run =(A, B). Allen sends A(x, 1, $\phi$) and Bella sends B(y, 1, $\phi$) in the beginning. Here, x and y represent a function of the party's input, which is Allen and Bella's input. 1 denotes a round number, and $\phi$ denotes a transcript. Two people cannot make a move at the same time in chess. When one makes a move, one must wait. For example, Allen makes the first move, followed by Bella, and so on. Therefore, the output of Allen is $\pi$A(x, n + 1,transA) and the output of Bella is $\pi$B(x, n + 1,transB) and here 'trans' is the transcript that is seen by Allen and Bella. We may see different outputs for Allen and Bella due to noisy channels. As we often used the term noise, we should know that there are three types of noise power: adversarial noise, computationally efficient noise, and random noise. With an adversarial noise rate of $\epsilon$, most n symbols are corruptible, where in computationally efficient noise, corruption is limited to a fraction of $\epsilon$-fractions of all transmissions. The symbols are disturbed independently of one another with random probability, creating what can be seen as a memoryless channel [4].

## III. Tree Codes

Tree code resembling a binary tree [6][7]. A possible infinite binary tree with edges labeled with sigma symbols. These edges are labeled sigma 1, 0 and so on, and their labeling satisfies the distance guarantee. From the root, let's

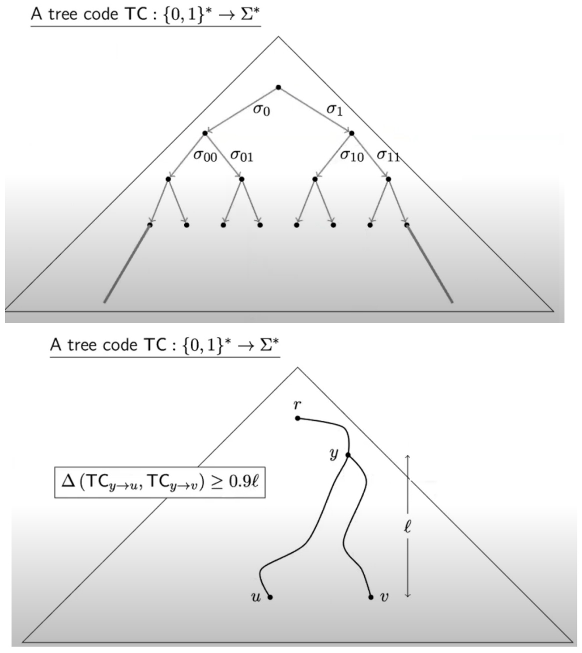
A tree code $\mathsf{TC}:\{0,1\}^*\to\Sigma^*$
$\sigma_0$
$\sigma_1$
$\sigma_{00}$
$\sigma_{01}$
$\sigma_{10}$
$\sigma_{11}$
A tree code $\mathsf{TC}:\{0,1\}^*\to\Sigma^*$
$r$
$y$
$\Delta(\mathsf{TC}_{y\to u},\mathsf{TC}_{y\to v})\geq 0.9\ell$
$\ell$
$u$
$v$

Fig. 4. Tree Code [6]

examine all labels on the path to the vertex, u and the vertex, v. As there is the same path from u to v before the least common ancestor, these labels must be the same as the least common ancestor but there is a large fraction 0.9 of all the locations where the labels from the least common ancestor y differ from those from the least common ancestor, u. As soon as you have transcripts that are different then in just a fraction of the locations the labels will be different from those from y to v, so if l is the length of the tags here, then 0.9 will be the length of the tags in this location. The problem is that with hash functions, you have a guarantee that for any two strings that are different, the scene is deterministic. It's not that simple because with the hash function, you get a deterministic scene. Using the tree codes to detect the different differences between strings, you detect them with a large constant probability. For strings that are different, the standardized version can only be obtained for 0.9 fraction of the two strings that are different [6][7]. Take a look at the algorithm below.

```
A finite alphabet, S;
if s = s1, s2, ...., sm and r = r1, r2, ...., m ⇒ $S$ then
    determine △(s, r);
    d, S, $\alpha, n$ end
```

**Algorithm 1:** Tree Code [5]

In this case, S is the finite alphabet, and s and r are the same-length words over S. For s and r and we need to determine the Hamming distance between them. A d-ary tree with a depth n has d children whose leaves are all at the depth n. $\alpha$ is the distance. Thus, d, S, $\alpha$ , and n are the four parameters of a tree code [5]. We can choose two nodes v1 and v2 that are at some common deepening h. Suppose W(v1) and W(v2) are the letters applied to the arcs connecting the root to v1 and v2, respectively. $\alpha$l should be the minimum distance between W(v1) and W(v2). $\alpha$ is fixed to ½ [5]. Tree codes have the key property that the alphabet size necessary for ensuring their existence does not depend upon n. A tree code with an infinite height exists for any fixed d constant size alphabet [5].

## IV. Results of Braverman's and Rao's algorithms

The days of error in simple transmission or two party communication are long gone. Now we know how to correct errors but is it possible to generalize those error correcting ideas to the setting of interactive communication? In this report we try to present exactly such a way. The main result of Braverman and Rao's paper is a new method of encoding protocols that can tolerate a large number of errors [1]. Given a two party communication protocol $\pi$, taking inputs from some finite set, it is possible to write $\pi$(x, y) to denote the transcript, namely the sequence of messages exchanged, when $\pi$ is run with inputs x, y. It is then possible to consider communication protocols that are allowed to send symbols from large alphabets. The paper wrote $|\pi|$ for the maximum number of symbols that can be exchanged on any setting of inputs, namely the communication complexity of $\pi$. Of course every communication protocol with a larger alphabet can be simulated by a protocol with a binary alphabet, but this translation can change the error rate that a protocol is resilient to. Thus the authors came up with two theorem as the final outcome.

*Theorem 4.1:* For every $\epsilon$, there is an alphabet $\Sigma$ of size O(1) and a transformation of communication protocols C$\epsilon$, such that for every binary protocol $\pi$, C$\epsilon$ ($\pi$) is a protocol using symbols from $\Sigma$, $|C\epsilon(\pi)| = O\epsilon(|\pi|)$, and for all inputs x, y, both parties can determine $\pi$(x, y) by running C$\epsilon$ ($\pi$) if the fraction of errors is at most 1/4 - $\epsilon$ [1].

*Theorem 4.2:* For every $\epsilon$, there is a transformation of communication protocols C$\epsilon$, such that for every binary protocol $\pi$, $C\epsilon(\pi)$ is a binary protocol, $|C\epsilon(\pi)| = O\epsilon(|\pi|)$, and for all inputs x, y, both parties can determine $\pi$(x, y) by running $C\epsilon(\pi)$ if the fraction of errors is at most 1/8 - $\epsilon$ [1].

## V. Encoding of Arbitrary Interactive Protocol

The main focus of the paper relies on how to encode an arbitrary interactive protocol so that recovery is possible when an adversary tries to mess up the transmission[1] and in order to demonstrate such a protocol the report relies on a protocol for solving a specific problem. But this problem is kind of complete for all communication protocols. So it'll be enough to talk about just this problem. So what's the problem? The problem is the pointer jumping problem.

### A. Pointer Jumping Problem

Let's have a full binary tree of depth N and the input here is a set of edges. So the, the first party knows all the 0 edges and the second party knows all the 1 edges. What are the 1 edges and 0 edges? So every Vertex at even height has exactly one 0 edge coming out of it while every vertex at odd height has exactly one 1 edge coming out of it. Here in this scenario the first party knows all of the 0 edges, and the second party knows all the 1 edges and their goal is to find the unique path that goes from the root of the tree down to a leaf. Ultimately there is exactly one such path. This problem

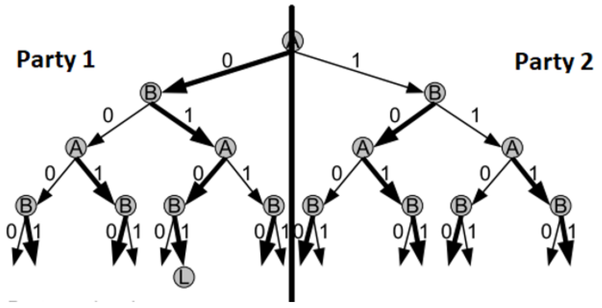

Fig. 5. : Full binary tree of depth N [1]

is complete for all communication protocols in the sense that if we have any protocol that communicates in bits and does something, then we can view that protocol as first taking the input, then viewing the input as a set of 0 edges and 1 edges, and then solving this problem. There is a trivial protocol to solve this problem with communication N which is, the first party will send a bit to indicate where the first edge goes. Then the second party will send just a single bit to indicate where the next edge goes. And in this way, they will find their goal after N bits of communication. Now the goal is going to be, to solve the same problem in the presence of errors. We want to find this 0, 1 path, but lots of symbols that are transmitted can be corrupted by an adversary and then can be received incorrectly. In each step, what the party does is it looks at the unique path in a union and extends it right. Other party then announces another edge to extend this path and so on. So why does not this protocol work when there are errors? If we think of it from the perspective of the first party, the first party knows the set of edges. This is what he announced. So there's no errors in that set. He doesn't know the set of edges announced by other party. And that's a problem. Because he announces something. Now the second party announced an edge, but it could be received by the first party as something different. So then the first party will start talking about something completely different. But it could be received by the second party as something that is consistent and they could kind of keep talking about completely random crazy parts of the tree in this way. In such scenario this protocol does not work.

### B. Recovery from Error

In this section we are going to discuss how to deal with error described in the previous section. The key object that is useful for dealing with these kinds of errors is something that Schulman invented, it is called a tree code. The only thing that is different in the work of Braverman and Rao is the way they used tree code in their paper.

### C. Error Correction

In this scenario party 0 and party 1 do not just send edges but rather they encode them using a tree code. Such an encoding allows them to eventually reconstruct the correct edges sent by the other side, despite the noise. A major property to be noted is that the parties never send incorrect information, even if one party guesses a wrong due to noise, the party always sends an edge that belongs to the same set. It can only be that the edge party 1 sends is not on the same set and thus it may be useless for party 2 and for the simulation. Yet, as time goes by, the tree code guarantees that party 2 correctly decodes a longer and longer prefix of the sequence of edges sent by party 1, and since this sequence always contains edges in Ex, then party 2 eventually will be able to decode the correct prefix of P0 and send an edge that continues that prefix. Note that the message of party 1 is of length O (log n) bits. Indeed, to extend a path, party 1 needs to indicate only how to proceed from the deepest edge it owns in that path. Since that edge was already communicated, say in the i-th transmission, party 1 can communicate the new edge by sending the pointer i and a path of length at most 2 descending from its i-th transmitted edge. Therefore, to support N rounds, we obtain the message space $M = [N](\{0, 1\}[\{0, 1\}2)$. The length of each message is thus logarithmic in the length of the protocol, but with a simple encoding it can become constant. It implies that if we run the protocol for over Epsilon steps, the protocol will succeed in the sense that if we run it for an Epsilon steps and we look at where the parties are consistent in the final round, they will be consistent to a point that includes all of the edges. The parties may not agree completely on what was announced, but they will agree about where the path goes in the original tree. So they will have concluded, found the right leaf tree.

## VI. Analysis

For analysis the authors have made a strong point that if the protocol did not identify the correct vertex v (X∪Y), then the number of errors in transmissions must have exceeded 2R • (1/4 - $Q$). They have also proved that if the first i rounds do not contain the first k edges on the correct path, then it must be the case that there is an inconsistency before the first k - 1 edges were announced. Ultimately the paper produced two main lemma[1].

*Lemma 6.1:* Error rate in [m(r) + 1, r] >= (1 - $\epsilon$) / 4

*Lemma 6.2:* r + 1 < t(k) means m(r) < t (k - 1)

## VII. Conclusion and Open Problem

The paper of Braverman and Rao is an improvement on Schulman's work in that it handles a larger (and potentially the maximum) fraction of errors. In addition, their scheme has no large hidden constants, which would make it in principle, practical if explicit efficient constructions of tree codes existed. To date, no constant-rate efficient constructions are known. There is a simple explicit construction, but it only achieves a rate of  (1/ log n) for trees of depth n. Thus we have been introduced to a few problems.

**Open Problem 1.** Construct an explicit constant-rate tree code with computationally efficient encoding and decoding.

**Open Problem 2.** Although the paper managed to show that robust protocols cannot tolerate a 1/4 error rate, it did not succeed in proving that arbitrary protocols must fail with this kind of error. So what is the maximum error rate that can be tolerated?

**Open Problem 3.** Is there a binary error-correcting scheme that recovers from a $\beta$-fraction of errors with $\beta \geq$ 1/8?

## Acknowledgements

We would like to thank Dr. Anand Kumar Narayanan for offering this course entitled 'Algebraic Coding Theory'. The topic is completely new for us, so it allows us to extend our knowledge in coding theory. It is our belief that the knowledge we gained from this course and project will be helpful to us in our future studies and works.

## References

[1] Braverman, Mark, and Anup Rao. "Toward coding for maximum errors in interactive communication." IEEE Transactions on Information Theory 60.11 (2014): 7248-7255.

[2] Fiche, Georges, and Gérard Hébuterne. Mathematics for engineers. John Wiley  Sons, 2013.

[3] Lecture 4: Noisy Channel Coding, Information-Theoretic Modeling. Teemu Roos Department of Computer Science, University of Helsinki

[4] Gelles, Ran. Coding for interactive communication: A survey. Now Publishers, 2017.

[5] Coding for Interactive Communication by Vijay Gupta, EE150, California Institute of Technology.

[6] Tutorial on Deterministic and Efficient Interactive Coding from Hard-to-Decode Tree Code by Raghuvansh R. Saxena.

[7] Brakerski, Zvika, Yael Tauman Kalai, and Raghuvansh R. Saxena. "Deterministic and Efficient Interactive Coding from Hard-to-Decode Tree Codes." 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 2020.