Depth-First Decoding of Distributed Arithmetic Codes for Uniform Binary Sources

Bowei Shan, Yong Fang, Chang'an University
Vladimir Stankovic, University of Strathclyde
Samuel Cheng, University of Oklahoma
En-hui Yang, University of Waterloo

Paper No. 162

1. INTRODUCTION

- Distributed Arithmetic Coding (DAC) :
  - A variant of the arithmetic coding (AC) that can be used to perform lossless distributed source coding
- Open Problems in DAC:
  - DAC's decoding complexity
  - How fast the complexity of the full-search DAC decoder grows with respect to code
- Previous Work (Fang et al. 13, Fang et al. 14, Fang et al. 15) :
  - Codebook Cardinality Spectrum (CCS)
  - Hamming distance (H-distance) spectrum (HDS)
  - Breadth-First Decoder of DAC (BFD)
- Drawbacks of BFD:
  - There is a risk that the optimal path is mis-pruned when its partial metric is inferior to other paths
  - To achieve good performance, a large amount of paths must be maintained during the decoding, which imposes a heavy burden on the decoder
- Contribution:
  - First realization of depth-first the DAC decoder
  - Experiments show that under the same complexity constraint, the depth-first decoder (DFD) outperforms the BFD, if the code is not too long and the SI quality is not very poor.

2. Review on Breadth-First DAC Decoder

- Problem Formulation
  - Assume that the source emits $X^n = x^n$, which is encoded at rate R to get M=m. If R<1, the SI Y^n=y^n that is correlated with x^n is necessary at the decoder for the lossless recovery of x^n. On receiving m, the decoder tries to find the binary vector best matching y^n from all solutions to $\lfloor 2^{nR} l(s^n) \rfloor = m$, where $s^n \in \mathbb{B}^n$. Then DAC decoding can be formulated as

$$\hat{x}^n = \arg\min_{s^n} d_H(s^n, y^n), \qquad s.t. \left\lfloor 2^{nR} l(s^n) \right\rfloor = m \qquad (1)$$

- Construction of DAC Tree
  - We define the following vector

$$u_i^j(s) \triangleq (u(s^i), \cdots, u(s^{j-1})) \qquad (2)$$

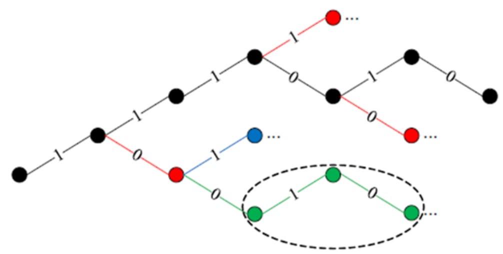
where i<j. If i=0, the subscript is dropped for simplicity. For boy nodes, i.e., $i \in [0:(n-t)]$, we have

- if $u(s^i) \in [0:(1-2^{-r}))$, node s^i has only 0-child
- if $u(s^i) \in [(1-2^{-r}), 2^{-r})$, node s^i has both 0-child and 1-child, which causes branching;
- if $u(s^i) \in [2^{-r}, 1)$, node s^i has only 1-child

For $i \in [(n-t):n)$, if $u(s^i) \in [0, 0.5]$ node s^i has only o-child; otherwise, node s^i has only 1-child. So there is no branching at tail nodes

3. Depth-First DAC Decoder

- Principle of Depth-First DAC Decoder:



- The principle of DFD can be illustrated by above figure

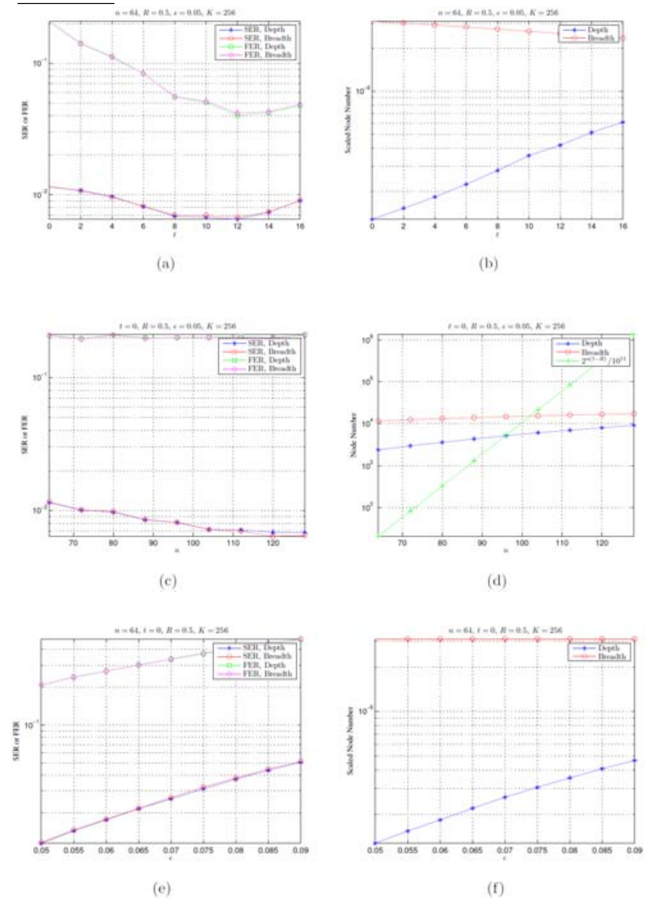
The SI is assumed to be $\overrightarrow{000010}$. The initial pass proceeds along the black full path $\overrightarrow{111010}$. After the initial pass, the decoder records the path-SI H-distance: $d_{\min} = d_H(\overrightarrow{100010}, \overrightarrow{111010}) = 2$. There are 3 fork nodes along the path $\overrightarrow{111010}$, so 3 unequal-length paths ($\overrightarrow{10}$, $\overrightarrow{1111}$, and $\overrightarrow{11100}$) are suspended, whose end nodes are marked with red color. Note that, branch picking/storing at fork nodes is based on overall path metrics rather than SI, so the 1-branch is selected at the first fork node, while the path $\overrightarrow{10}$ is suspended. After the initial pass, the decoder selects the best, i.e., with the greatest overall metric, suspended path $\overrightarrow{10}$ to trigger a new pass. The second pass proceeds along the path and is early aborted because the path-SI H-distance $d_H(\overrightarrow{100010}, \overrightarrow{10010*}) \geq d_{min} = 2$. During the second pass, the path $\overrightarrow{101}$ is suspended, whose end node is marked with blue color. Note that, the memory allocated for the early-aborted path $\overrightarrow{10010}$ can be partially released because the last three nodes (marked with green color) $\overrightarrow{10010}$ solely belong to the path.

- Pseudo Code for the DFD of DAC

```
function depth_first_dac_decoder(u_0)
    s ← create_root(u_0)
    d_min ← n
    while the termination condition is not satisfied do
        isFull ← pass(s, d_min)
        if isFull = true then
            d_min ← bst.d
            compact_list(spaths, d_min)
        end if
        s ← wakeup_path(spaths)
    end while
    x̂^n ← trace_back(bst)
end function
```

4. Experimental Results

- Experimental Results Demonstrate How Tail Length, Code Length, and SI Quality Impact the DFD and the BFD that are Subject to Equivalent Constraints.



(a) (b) (c) (d) (e) (f)

5. CONCLUSION AND SUMMARY

- This research work presents a depth-first decoding algorithm for distributed arithmetic codes under uniform binary sources.
- The DFD's complexity can be lowered by enhancing the SI quality: The better SI, the lower complexity.
- Compared with the BFD, the DFD performs better for short and medium code lengths and in situations when SI quality is not too poor.