Trellis Code Design for Combined Channel Estimation and Error Correction

Huei-Ya Chang

Advisor: Prof. Po-Ning Chen

Institute of Communications Engineering,

National Chiao-Tung University

Outline

- Introduction
- Technical Background
- Trellis Code Design for Block Fading Channels
- Decoding Method
- Simulation Results and Future Work

Outline

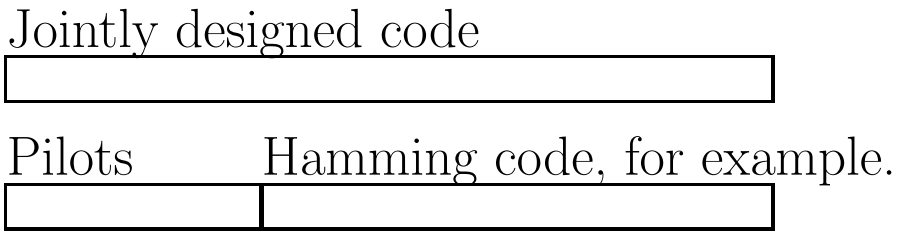
- Introduction
- Technical Background
- Trellis Code Design for Block Fading Channels
- Decoding Method
- Simulation Results and Future Work

Introduction

- In a typical communication system, the receiver will perform **channel estimation**, **channel equalization** and **error correction** separately in sequence.
- In 2002, Skoglund *et al.* confirmed that the computer searched code could outperform the traditional separately designed system with perfect channel estimation by 2 dB under quasi-static multipath block fading.
- This, as well as other recent researches, suggests that jointly considering these system devices should be a promising system design.

Introduction

- As an emphasization, we do not send the pilots or training sequence.
- **All** the transmission energies are used to transmit the "information-bearing bits."
- The channel estimation can be regarded as being done implicitly using the "information-bearing bits."



- In 2006, Wu *et al.* proposed a systematic construction of the **self-orthogonal tree-structure** (SOTS) codes for combined channel estimation and error correction.
  - These codes have almost the same performance as the best computer-searched codes.
  - Yet, the decoding complexity still grows exponentially with the code word length because of its tree structure.
- Our target is to design a trellis code for combined channel estimation and error correction in order to reduce the decoding complexity.

Outline

- Introduction
- Technical Background
- Trellis Code Design for Block Fading Channels
- Decoding Method
- Simulation Results and Future Work

Block Fading Channel Model

A single-input-single-output (SISO) time-discrete system can be modelled as

$$\boldsymbol{y} = \mathbb{B}\boldsymbol{h} + \boldsymbol{n},$$

where

$$\mathbb{B} \triangleq \begin{bmatrix} b_1 & 0 & \cdots & 0 \\ \vdots & b_1 & \ddots & \vdots \\ b_N & \vdots & \ddots & 0 \\ 0 & b_N & \ddots & b_1 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & b_N \end{bmatrix}_{L \times P}.$$

- $\boldsymbol{b} = [b_1, \ldots, b_N]^T$ is the transmitted codeword.
- $\mathbb{B}$ emulates the convolution operation with channel coefficient $\boldsymbol{h}$.
- $b_i \in \{-1, +1\}$, where $1 \leq i \leq N$.
- $\boldsymbol{h}$ is the $P \times 1$ channel coefficient vector and remains constant within a coding block with length $L = N + P - 1$.
- $\boldsymbol{n}$ is the $L \times 1$ zero-mean complex Gaussian noise vector with covariance matrix $\sigma^2 \mathbb{I}_L$.

Since $\boldsymbol{h}$ is unknown and we assume the transmitted codeword matrix $\mathbb{B}$ is equal likely, the joint maximum-likelihood (JML) detection becomes the best option:

$$(\hat{\mathbb{B}}, \hat{\boldsymbol{h}}) = \arg \max_{\mathbb{B}} \max_{\boldsymbol{h}} \Pr(\boldsymbol{y}|\mathbb{B}, \boldsymbol{h}).$$

For a given $\mathbb{B}$, the channel coefficient $\boldsymbol{h}$ that maximizes $\Pr(\boldsymbol{y}|\mathbb{B}, \boldsymbol{h})$ can be pre-determined, i.e.,

$$\hat{\boldsymbol{h}} = \arg \max_{\boldsymbol{h}} \Pr(\boldsymbol{y}|\mathbb{B}, \boldsymbol{h}).$$

Then, the JML estimate of the transmitted codeword is

$$\begin{aligned} \hat{\boldsymbol{b}} &= \arg \max_{\boldsymbol{b}} \Pr(\boldsymbol{y}|\mathbb{B}, \hat{\boldsymbol{h}}) \\ &= \arg \min_{\boldsymbol{b}} \|\boldsymbol{y} - \mathbb{B}(\mathbb{B}^H\mathbb{B})^{-1}\mathbb{B}^H\boldsymbol{y}\|^2 \\ &= \arg \min_{\boldsymbol{b}} \|\boldsymbol{y} - \mathbb{P}_B\boldsymbol{y}\|^2, \end{aligned}$$

where $\mathbb{P}_B \triangleq \mathbb{B}(\mathbb{B}^H\mathbb{B})^{-1}\mathbb{B}^H$. Noted that $\mathbb{B}$ and $\mathbb{P}_B$ are not one to one correspondence unless the first bit b_1 is fixed as, for example, -1.

The JML Detection

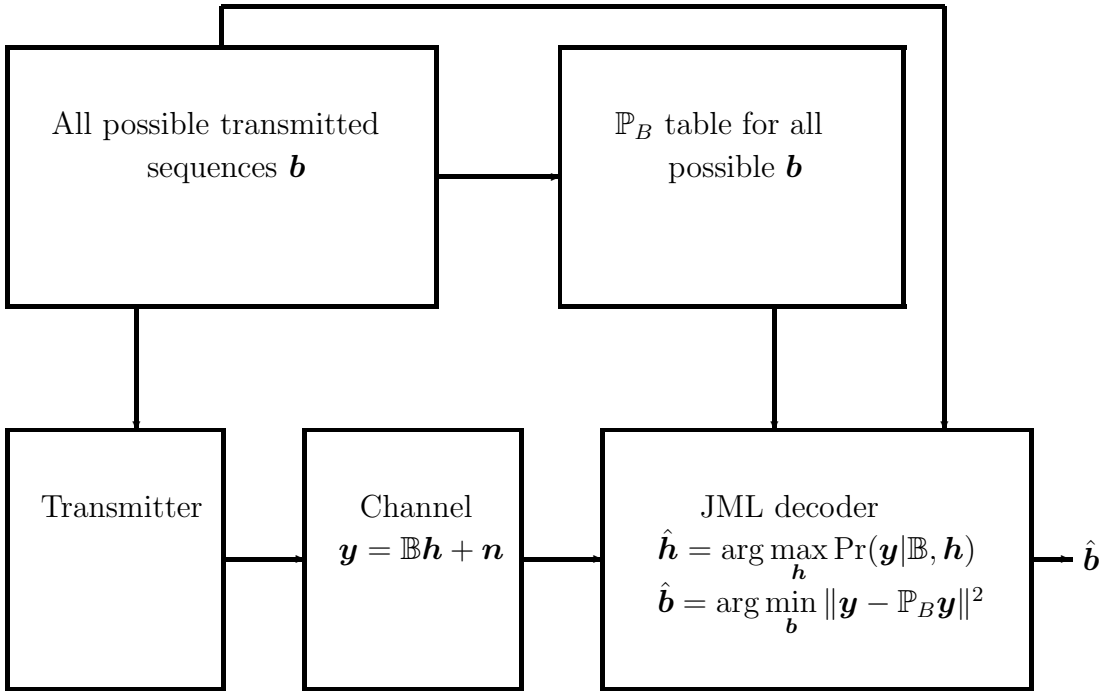


Figure 1: An Illustrator of JML.

The SOTS Code

- $\mathbb{G}_\theta \triangleq \begin{bmatrix} N & \theta \\ \theta & N \end{bmatrix}$ for $\theta \in \{-1, 0, +1\}$
- The average **SNR** can be optimized if the codeword is self-orthogonal to its shift counterpart, i.e.

$$\frac{1}{N}\mathbb{B}^H\mathbb{B} = \mathbb{I}_P$$

  where $\mathbb{I}_P$ is the $P \times P$ identity matrix.
- When N is even, $\mathbb{I}_P$ must be relaxed to $\frac{1}{N}\mathbb{G}_\theta$.

The SOTS Code

- The codeword selection process of SOTS codes is to list all the candidate sequences (i.e., self-orthogonal sequences) in their binary alphabetical order and pick the codewords from the ordered list in a fixed interval.

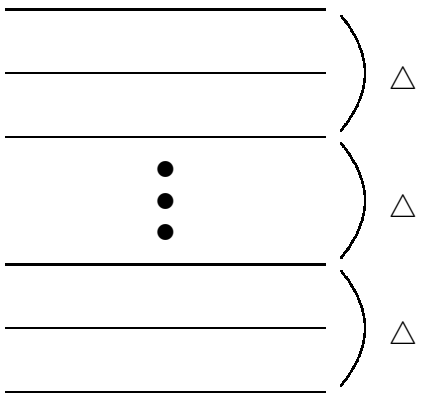


Figure 2: The selection process of SOTS code

Outline

- Introduction
- Technical Background
- Trellis Code Design for Block Fading Channels
- Decoding Method
- Simulation Results and Future Work

Principles of Design

The principles of code design:

- The code needs to be in a trellis structure.
- The state number at each level cannot exceed an upper limit S.
- The self-orthogonality of codeword is maintained.
- The distance among $\mathbb{P}_B$ in the $\mathbb{P}_B$ domain is made as large as possible.
- The performance is acceptable.

Our goal is to fit the SOTS code into a trellis structure in order to reduce the growing decoding complexity with respect to the codeword length.

Overview of Trellis Code Construction Algorithm

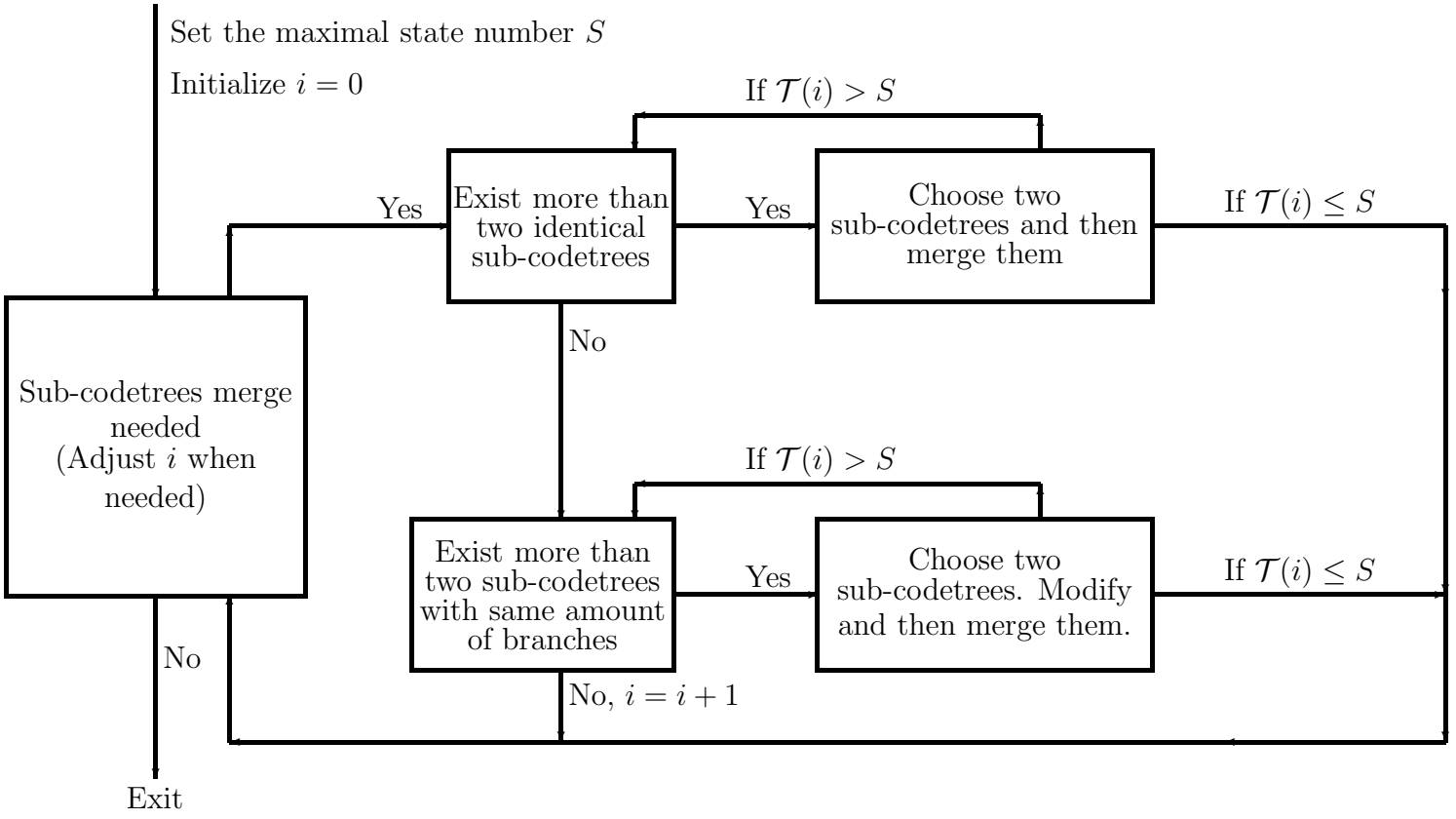


Figure 3: An Illustrator of Trellis Code Construction Algorithm

Check Whether Sub-Codetrees Merge is Necessary

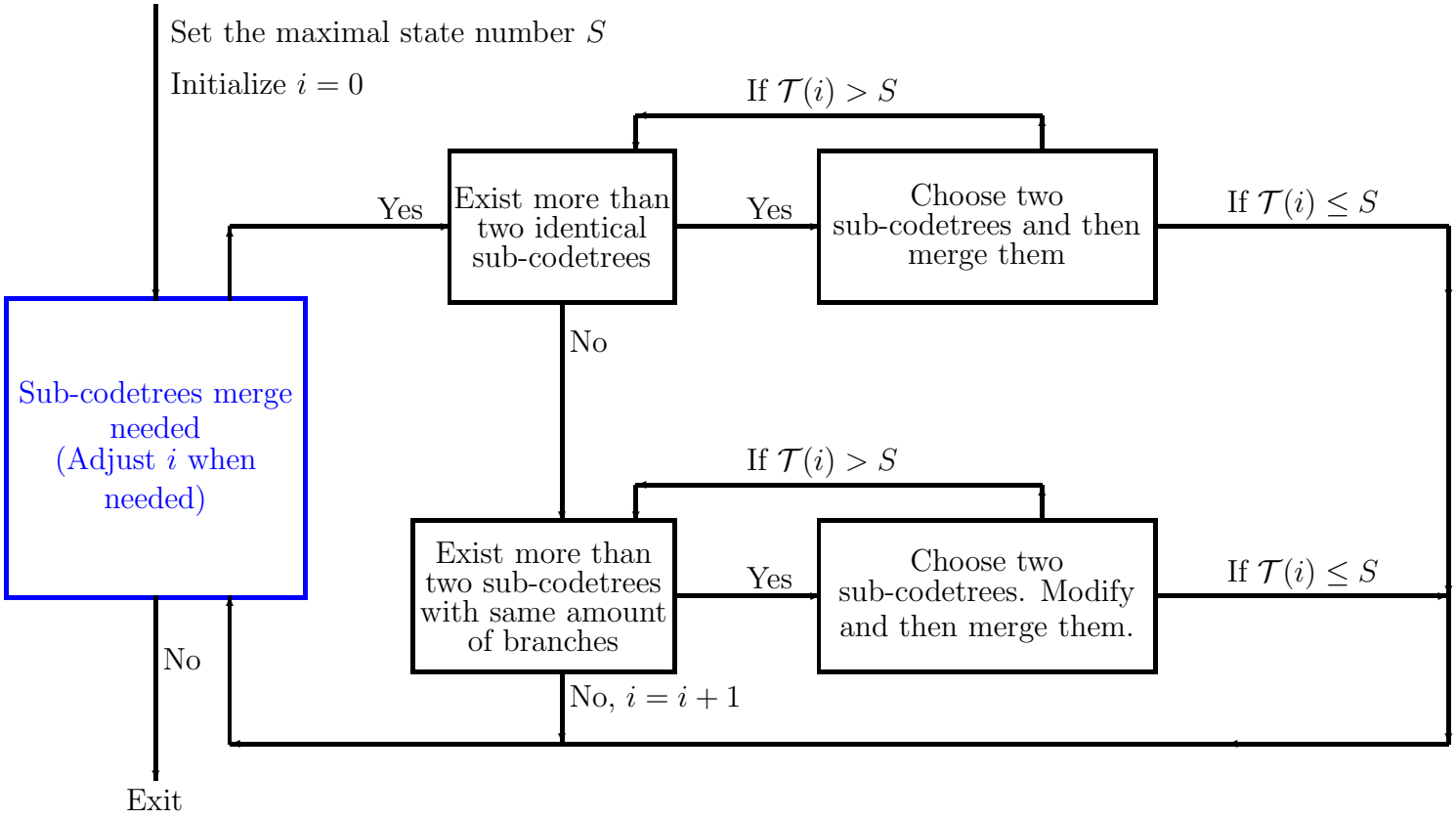


Figure 4: An Illustrator of Trellis Code Construction Algorithm

Check Whether Sub-Codetrees Merge is Necessary

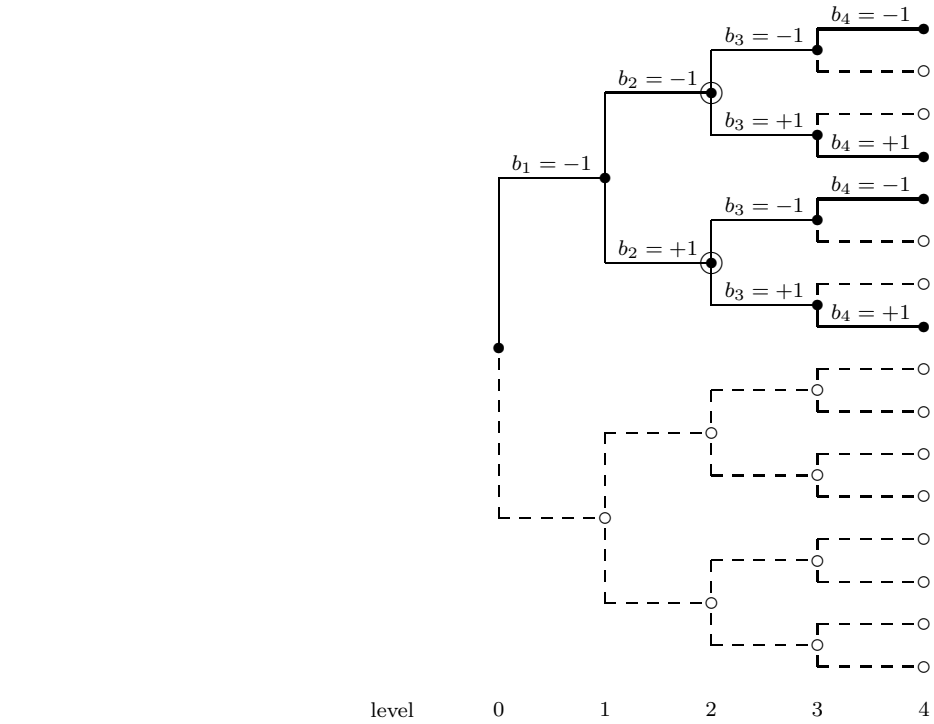


Figure 5: Example of a code tree with b_1 fixed as -1.

- $\mathcal{T}(i)$ is the set that consists of all sub-codetrees with root nodes at level i.

Check Whether Sub-Codetrees Merge is Necessary

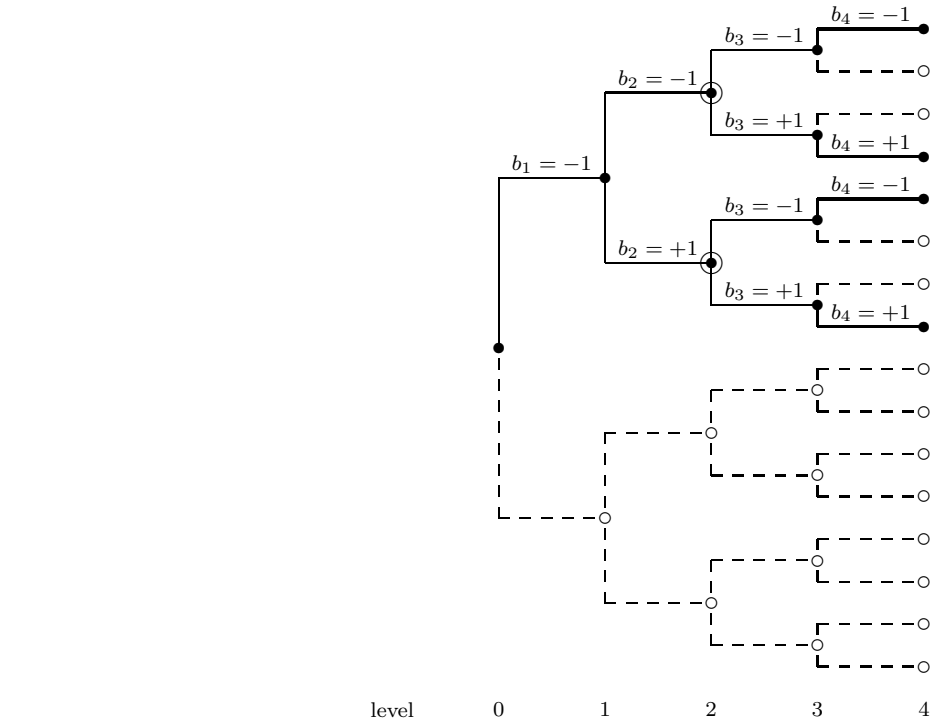


Figure 4: Example of a code tree with b_1 fixed as -1.

Step 1. Initialize level $i = 0$. Set the maximal state number S.

Step 2. . If $|\mathcal{T}(i)| > S$, merge sub-codetrees until $|\mathcal{T}(i)| \leq S$ or until there is no more sub-codetrees pairs that can be merged.

Step 3. if $i < N$, then set $i = i + 1$ and go to Step 2; else, stop the algorithm.

Sub-codetrees Merge: Identical Sub-codetrees Exist

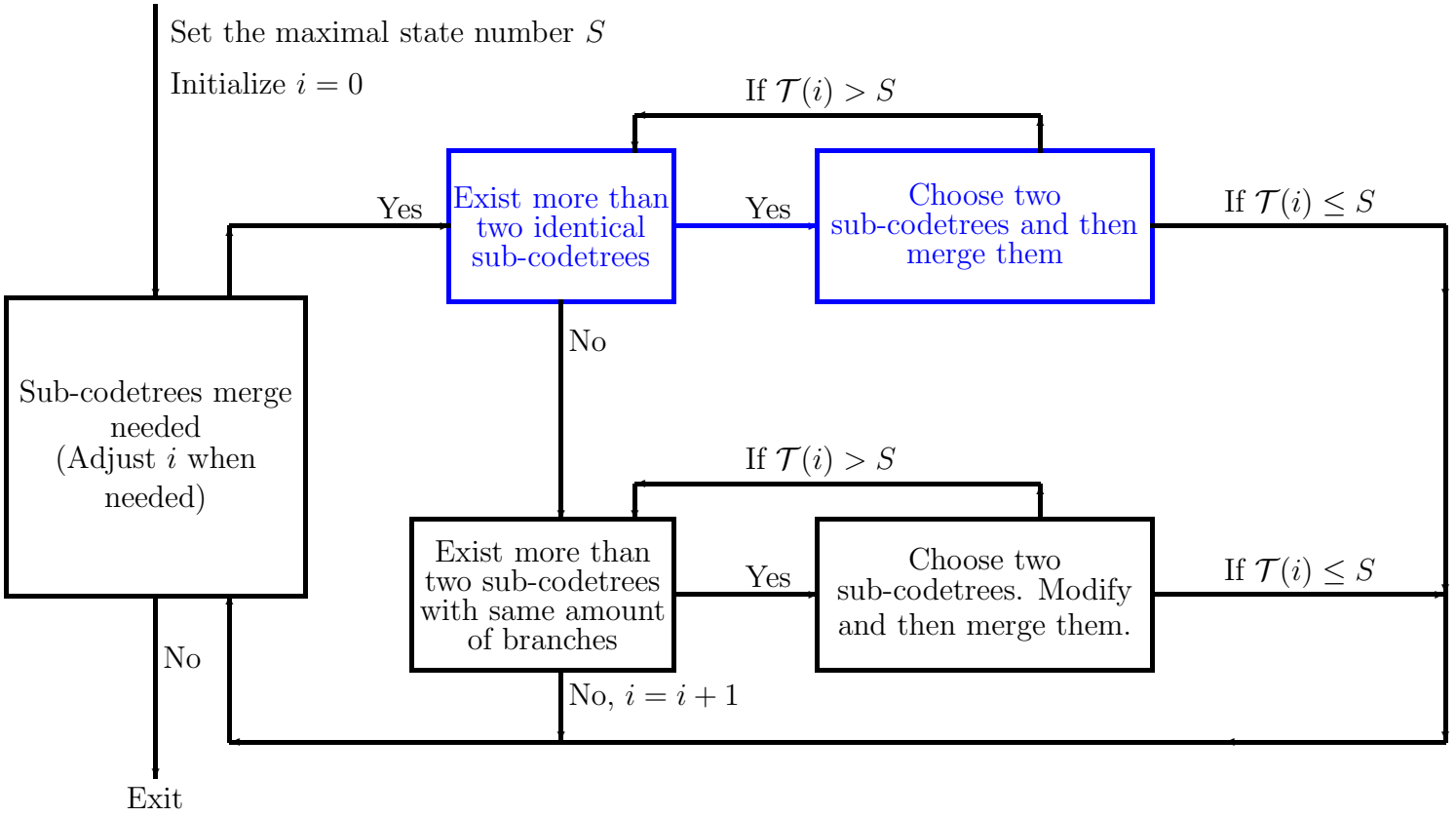
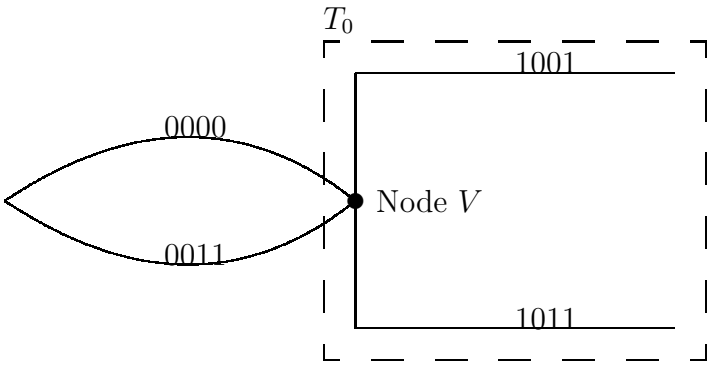


Figure 6: An Illustrator of Trellis Code Construction Algorithm

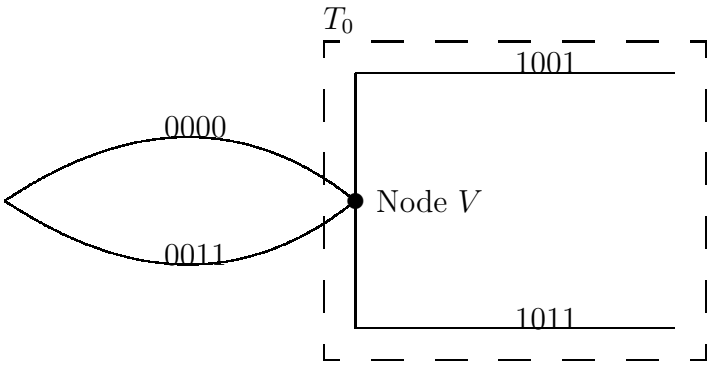
Sub-codetrees Merge: Identical Sub-codetrees Exist



Notations

- $\boldsymbol{a}_{(\ell)} = [a_1, a_2, \ldots, a_\ell]$ is the path entering a node at level ℓ.
- $\tilde{\boldsymbol{a}}_{(\ell+1)} = [a_{\ell+1}, a_{\ell+2}, \ldots, a_N]$ is the path from a node at level ℓ to a terminal node.
- V is the smallest decimal value of paths entering to the root of sub-codetree T_V. Here, $V = \min\{(0000)_2, (0011)_2\} = 0$.
- $N_{V,\tilde{\boldsymbol{a}}_{(\ell+1)}}$ is the number of code paths $\boldsymbol{a} = [a_1, a_2, \ldots, a_N]$ whose last $(N - \ell)$ bits are equal to $\tilde{\boldsymbol{a}}_{(\ell+1)}$ in sub-codetree T_V. Here, $N_{0,[1001]} = N_{0,[1011]} = 2$.

Sub-codetrees Merge: Identical Sub-codetrees Exist



- $\mathcal{E}_V$ and $\tilde{\mathcal{E}}_V$ are respectively the sets containing all code path portions ending at and originating from node V.

  $\mathcal{E}_V = \{0000, 0011\}$ and $\tilde{\mathcal{E}}_V = \{1001, 1011\}$.

- $\overline{\mathcal{O}}_V$ be the set that contains all binary sequence $\tilde{\boldsymbol{a}}_{(\ell+1)}$ satisfying $\mathbb{B}^T\mathbb{B} = \mathbb{G}_\theta$ for some $\boldsymbol{a}_{(\ell)}$ in $\mathcal{E}_V$. Here, $\overline{\mathcal{O}}_V = \{0101, 1001, 1011, 1101\}$ for $\theta = 1$ and $\boldsymbol{a}_{(4)} = \{0000\}$.

Sub-codetrees Merge: Identical Sub-codetrees Exist

If there exist more than two identical sub-codetrees in $\mathcal{T}(i)$,

then merge the two with decimal values V_1 and V_2 that maximize h-function value,

where

$$h(V_1, V_2) \triangleq \frac{|\overline{\mathcal{O}}_{V_1}| + |\overline{\mathcal{O}}_{V_2}|}{\max\limits_{\tilde{\boldsymbol{a}}_{(\ell+1)} \in \tilde{\mathcal{E}}_{V_1}} N_{V_1, \tilde{\boldsymbol{a}}_{(\ell+1)}} + \max\limits_{\tilde{\boldsymbol{a}}_{(\ell+1)} \in \tilde{\mathcal{E}}_{V_2}} N_{V_2, \tilde{\boldsymbol{a}}_{(\ell+1)}}}.$$

Sub-codetrees Merge: Identical Sub-codetrees Exist

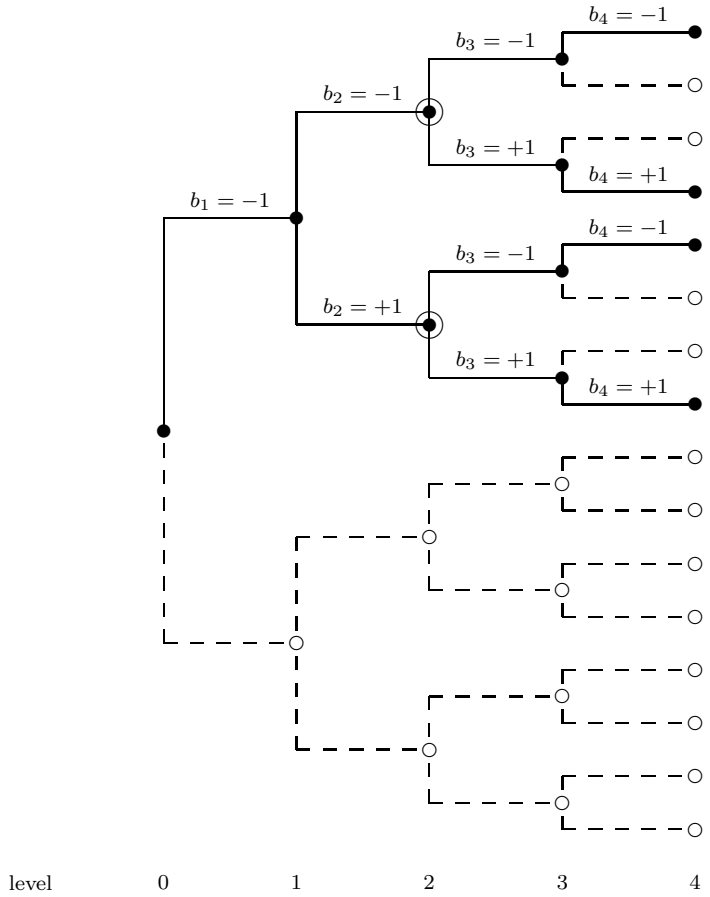


Figure 7: Example of a code tree with b_1 fixed as -1.

Sub-codetrees Merge: Identical Sub-codetrees Exist

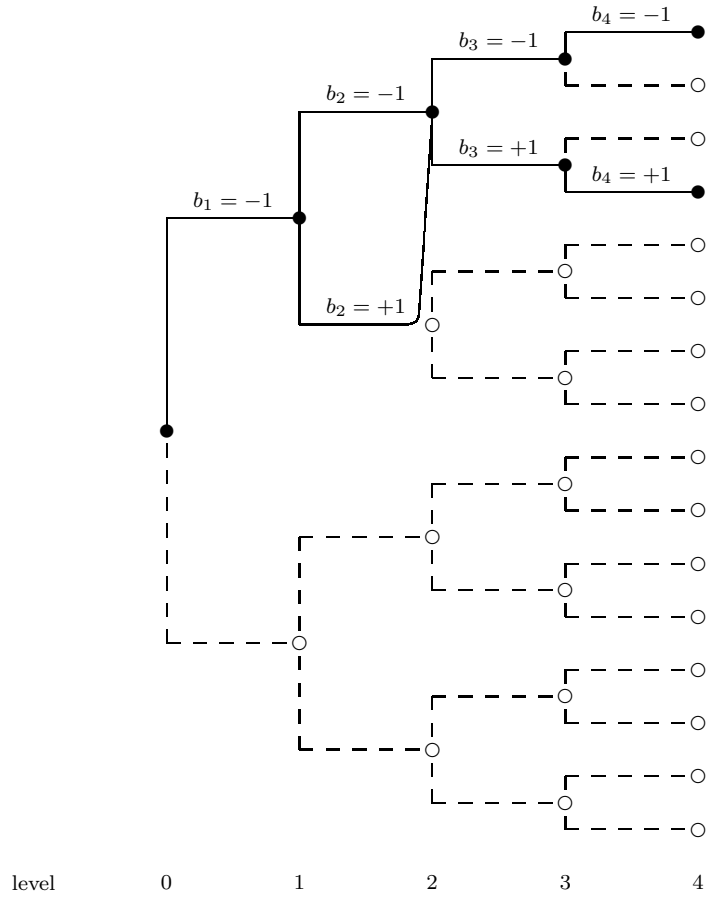


Figure 8: The trellis obtained by merging identical sub-codetrees in Fig. 7

Sub-codetrees Merge: No Identical Sub-codetrees

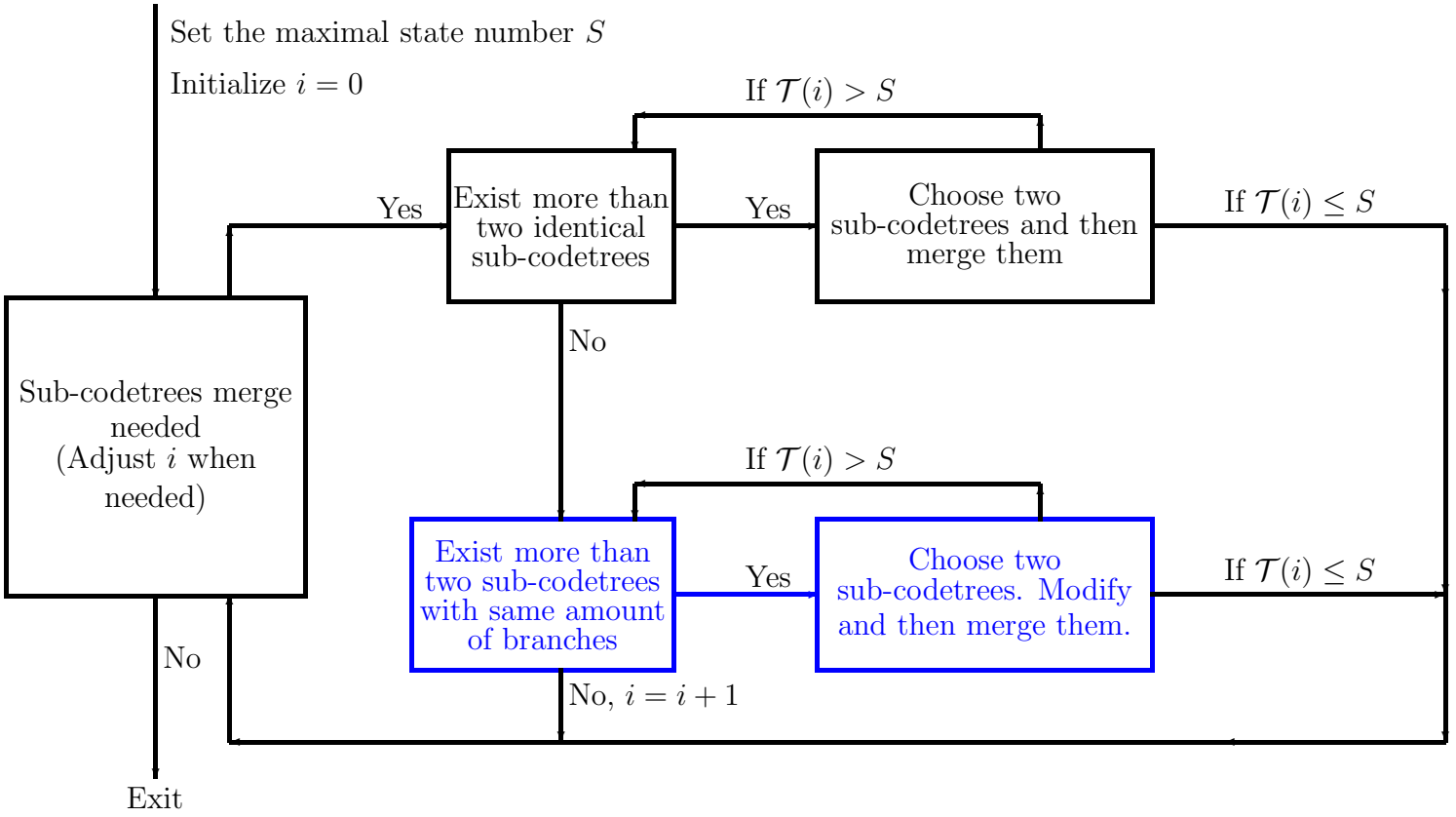


Figure 9: An Illustrator of Trellis Code Construction Algorithm

Sub-codetrees Merge: No Identical Sub-codetrees

Once there exists no identical sub-codetree pair,

then we choose the two sub-codetrees T_{V_1} and T_{V_2} that maximize $\hat{h}$-function to merge, where

$$\hat{h}(V_1, V_2) \triangleq \frac{|\tilde{\mathcal{O}}_{V_1} \cap \tilde{\mathcal{O}}_{V_2}|}{1 + |D_{V_1,1} - D_{V_2,1}|}.$$

More notations:

- $\tilde{\mathcal{O}}_V$ is the set containing all binary sequence $\tilde{\boldsymbol{a}}_{(\ell+1)}$ satisfying $\mathbb{B}^T\mathbb{B} = \mathbb{G}_\theta$ for every $\boldsymbol{a}_{(\ell)}$ in $\mathcal{E}_V$.
- $D_{V,I}$ is the decimal value of the I-th path in sub-codetree T_V.

Sub-codetrees Merge: No Identical Sub-codetrees

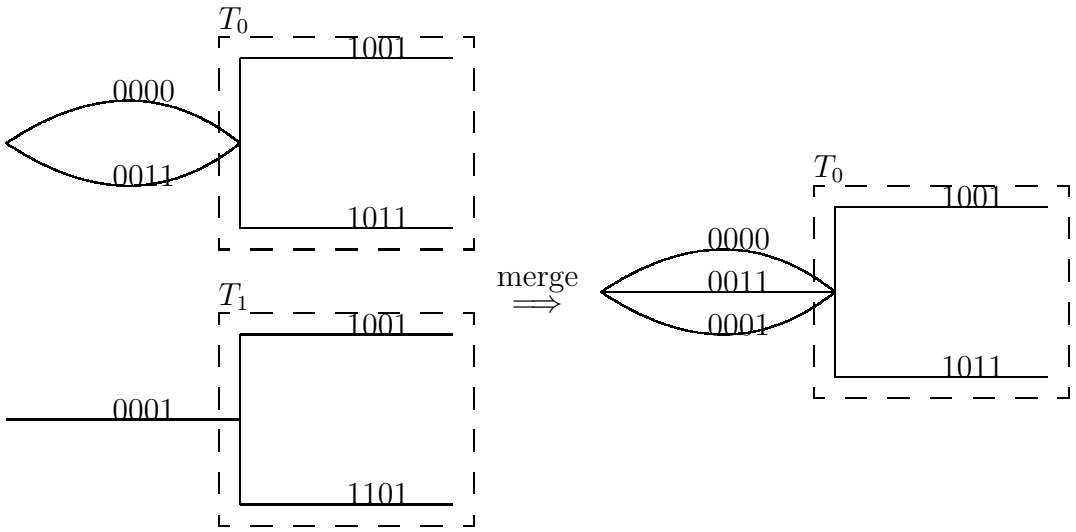


Figure 10: Illustration of the merging process.

- $I_{V,\tilde{\boldsymbol{a}}_{(\ell+1)}}$ denotes the ordered index of path $\tilde{\boldsymbol{a}}_{(\ell+1)}$ in sub-codetree T_V. Here, $I_{0,[1001]} = 1$, $I_{0,[1011]} = 2$, $I_{1,[1001]} = 1$ and $I_{1,[1101]} = 2$.
- Recall that $N_{0,[1001]} = N_{0,[1011]} = 2$ and $N_{1,[1001]} = N_{1,[1101]} = 1$.
- We then base on $\mathcal{E}_0 = \{[0000], [0011]\}$ and $\mathcal{E}_1 = \{[0001]\}$ to get that $\tilde{\mathcal{O}}_0 \cap \tilde{\mathcal{O}}_1 = \{[1001], [1011], [1101]\}$.

Sub-codetrees Merge: No Identical Sub-codetrees

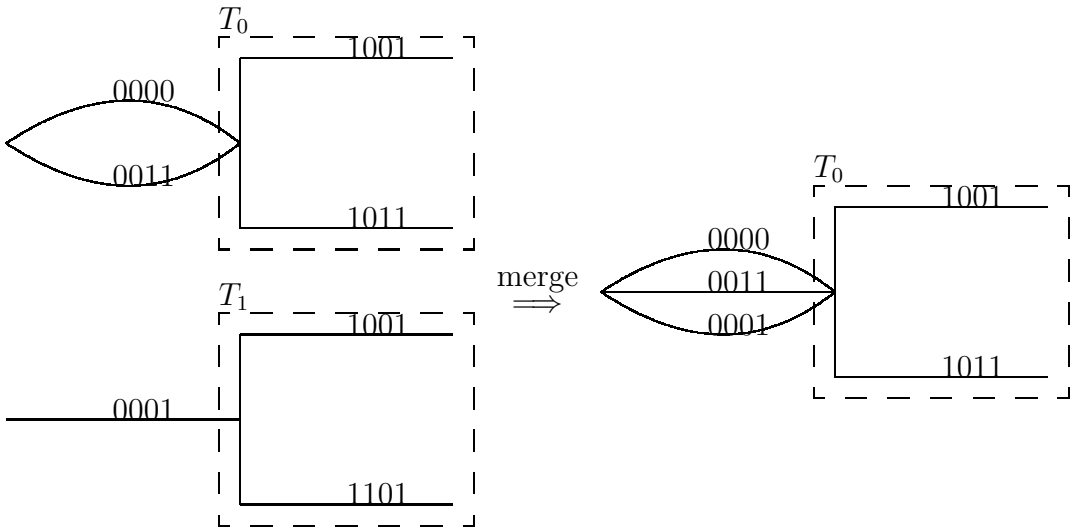


Figure 11: Illustration of the merging process.

- Next, we merge the two sequences with the same ordered index
  - For the first ordered index, $\tilde{a}_{(5)} = [1001]$ as there is no conflict.
  - However, for the second ordered index, which one should be used, [1011] or [1101]?

Sub-codetrees Merge: No Identical Sub-codetrees

– We will use the sequence in $\tilde{\mathcal{O}}_0 \cap \tilde{\mathcal{O}}_1$ whose decimal value is closest to weighted average W_2, which is [1011] in this example, where

$$W_2 = \frac{N_{0,[1011]} \cdot \mathsf{D}([1011]) + N_{1,[1101]} \cdot \mathsf{D}([1101])}{N_{0,[1011]} + N_{1,[1101]}} = \frac{2 \cdot 11 + 1 \cdot 13}{2 + 1} = 11.7$$

and $\mathsf{D}(\cdot)$ is a mapping function that simply returns the decimal value of a binary sequence $\boldsymbol{a}_{(\ell)} = [a_1, a_2, \ldots, a_\ell]$.

We summarize the Trellis Code Design Algorithm in the next slide.

Trellis Code Design Algorithm

Step 1. Initialize level $i = 0$. *Set the maximal state number* S.

Step 2. If $|\mathcal{T}(i)| \leq S$, *go to Step 7.*

Step 3. If there exist more than two identical sub-codetrees in $\mathcal{T}(i)$, *merge the two with decimal values* V_1 *and* V_2 *that maximize h-function value among all identical sub-codetree pair in* $\mathcal{T}(i)$ *and go to Step 2.*

Step 4. If $|\mathcal{T}(i)| \leq S$, *go to Step 7.*

Step 5. If there exist sub-codetree pairs in $\mathcal{T}(i)$ *satisfying* $|\tilde{\mathcal{E}}_{V_1}| = |\tilde{\mathcal{E}}_{V_2}|$, *merge the two that maximize* $\hat{h}$*-function value among all equal-*$|\tilde{\mathcal{E}}_V|$*-value sub-codetree pair in* $\mathcal{T}(i)$*; else, go to Step 7.*

Step 6. For the two merged sub-codetrees T_{V_1} *and* T_{V_2} *in Step 5, calculate*

$$W_s = \frac{N_{V_1,s} \cdot D_{V_1,s} + N_{V_2,s} \cdot D_{V_2,s}}{N_{V_1,s} + N_{V_2,s}}$$

for $1 \leq s \leq |\tilde{\mathcal{E}}_{V_1}|$. *Then, adjust*

$$\tilde{\boldsymbol{a}}(V_1, s) = \tilde{\boldsymbol{a}}(V_2, s) = \arg \min_{\tilde{\boldsymbol{a}}_{(i+1)} \in \tilde{\mathcal{O}}_{V_1} \cap \tilde{\mathcal{O}}_{V_2}} \left| \mathsf{D}\left(\tilde{\boldsymbol{a}}_{(i+1)}\right) - W_s \right|$$

for $1 \leq s \leq |\tilde{\mathcal{E}}_{V_1}|$. *Go to Step 4.*

Step 7. if $i < N$, $i = i + 1$ *and go to Step 2; else, stop the algorithm.*

Outline

- Introduction
- Technical Background
- Trellis Code Design for Block Fading Channels
- Decoding Method
- Simulation Results and Future Work

Step 1. Load the stack with the path that ends at the original node.

Step 2. Evaluate the metric values of the successor paths of the current top path in the stack. Then delete this top path from the stack.

Step 3. If the successor paths end at the same node as some paths already in the stack as well as the other successor paths, delete all of them except the one with least metric value.

Step 4. If there are undeleted successor paths after Step 3, insert these undeleted successor paths into the stack such that the paths in the stack are ordered according to their ascending metric values.

Step 5. If the top path in the stack ends at a terminal node in the code trellis, output the labels corresponding to the top path, and stop the algorithm; otherwise, go to Step 2.

Outline

- Introduction
- Technical Background
- Trellis Code Design for Block Fading Channels
- Decoding Method
- Simulation Results and Future Work

Simulation Results

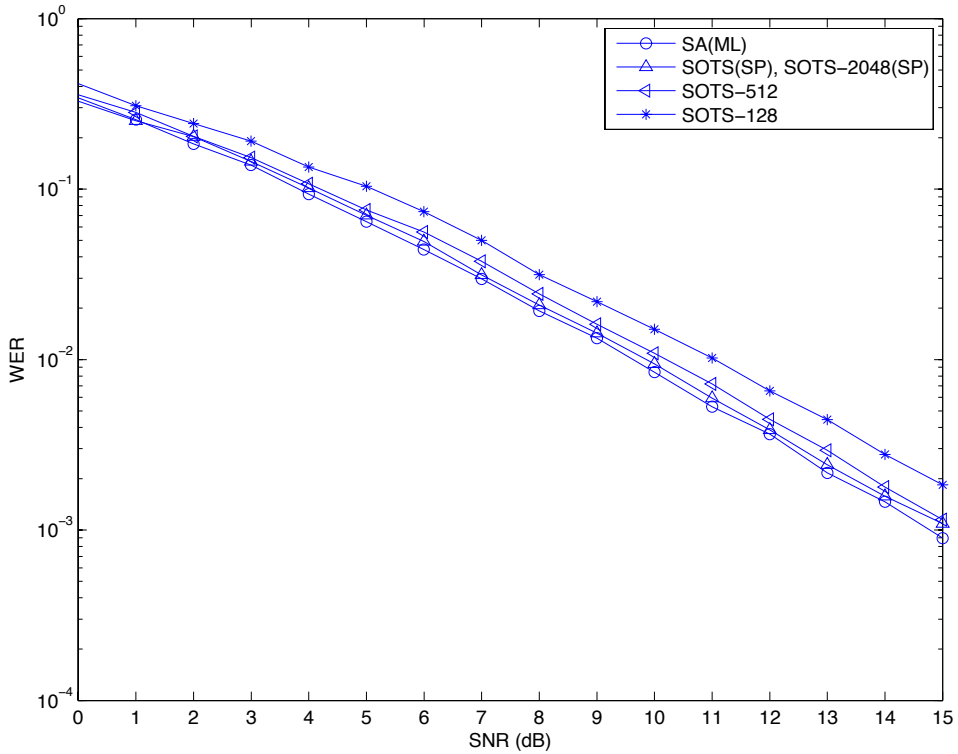


Figure 12: Word error rates (WERs) for the computer-searched code found by simulated annealing (SA(ML)), the SOTS code with single code tree (SOTS(SP)) which is also the code with maximum state number 2048 (SOTS-2048(SP)), the code with maximum state number 512 (SOTS-512(SP)), and the code with maximum state number (SOTS-128(SP)). The SA(ML) code is decoded by ML decoder , while all the other codes are decoded by the SP decoder. The code rate is $1/2$ and the codeword length is $N = 22$.

Simulation Results

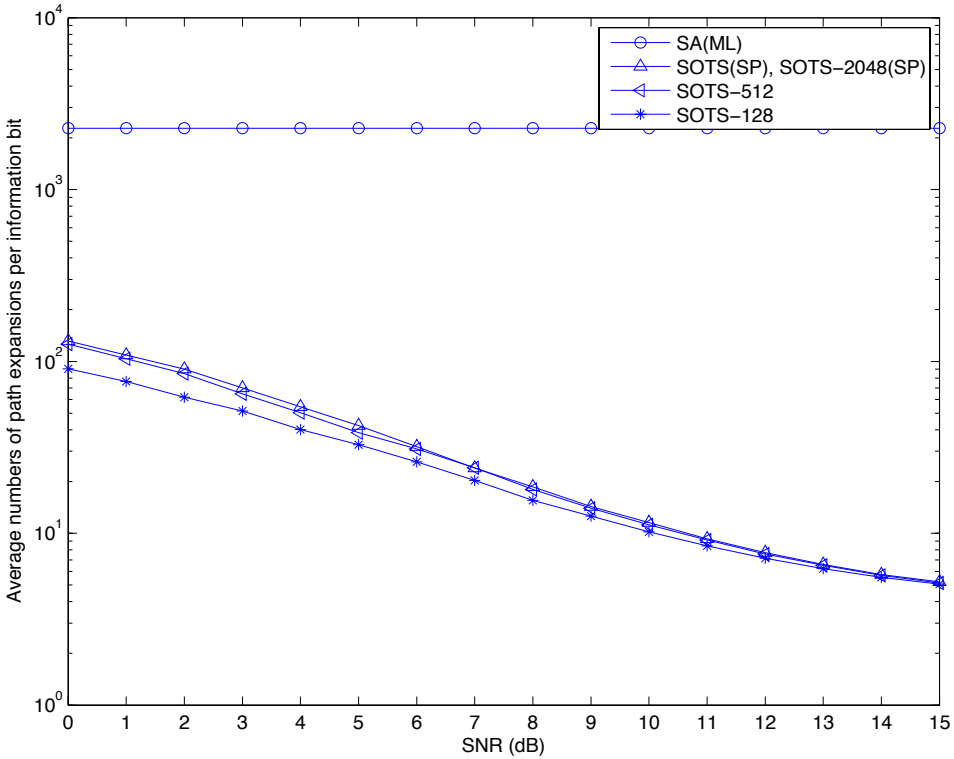


Figure 13: The average numbers of path expansions per information bit for the computer-searched code found by simulated annealing (SA(ML)), the SOTS code with single code tree (SOTS(SP)) which is also the code with maximum state number 2048 (SOTS-2048(SP)), the code with maximum state number 512 (SOTS-512(SP)), and the code with maximum state number (SOTS-128(SP)). The SA(ML) code is decoded by ML decoder , while all the other codes are decoded by the SP decoder. The code rate is 1/2 and the codeword length is $N = 22$.

Simulation Results

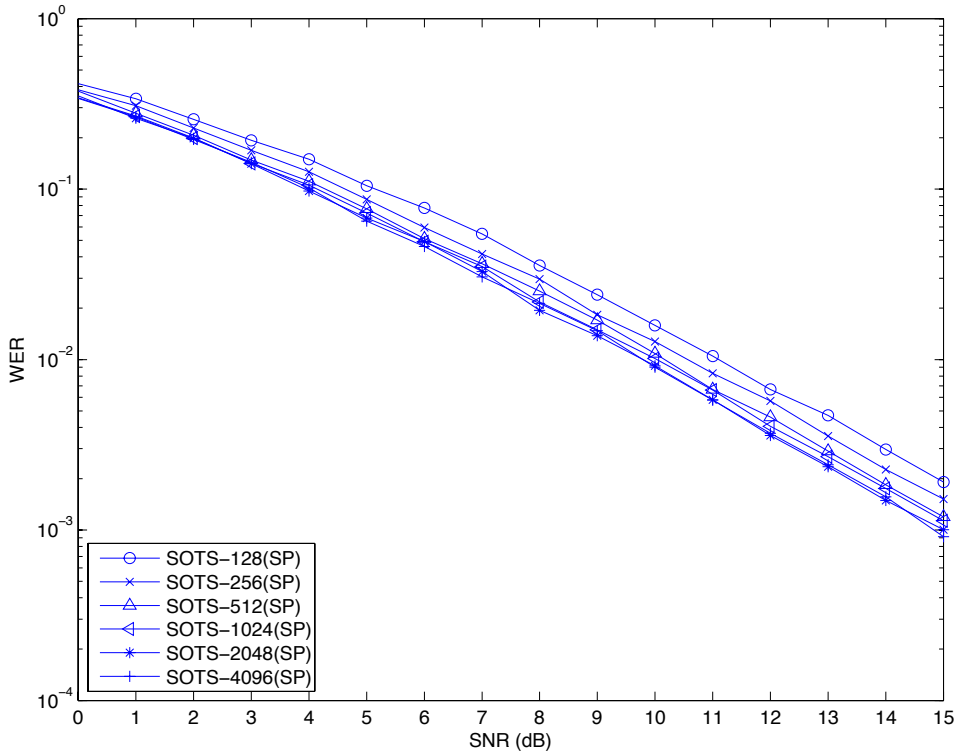


Figure 14: Word error rates (WERs) for the self-orthogonal trellis-structure codes of length $N = 24$ with maximal state number 128, 256, 512, 1024, 2048 and 4096, respectively. The decoders used are the sequential decoding with heuristic prediction (SP). The code rate is 1/2.

Simulation Results

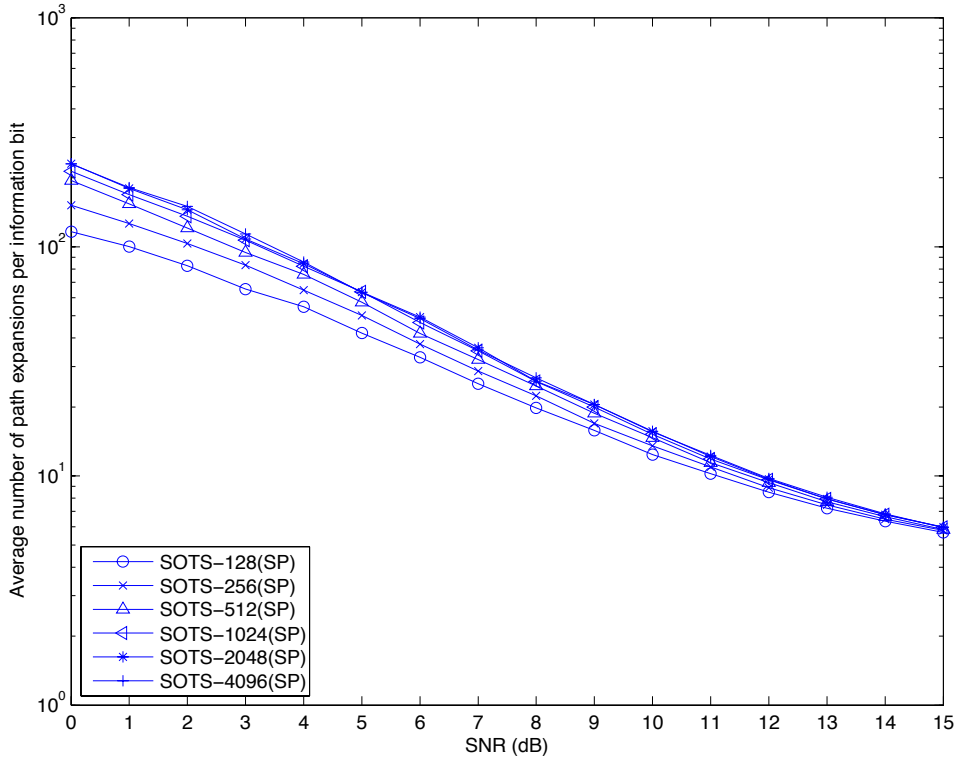


Figure 15: The average numbers of path expansions per information bit for the self-orthogonal trellis-structure codes of length $N = 24$ with maximal state number 128, 256, 512, 1024, 2048 and 4096, respectively. The decoders used are the sequential decoding with heuristic prediction (SP). The code rate is 1/2.

Simulation Results

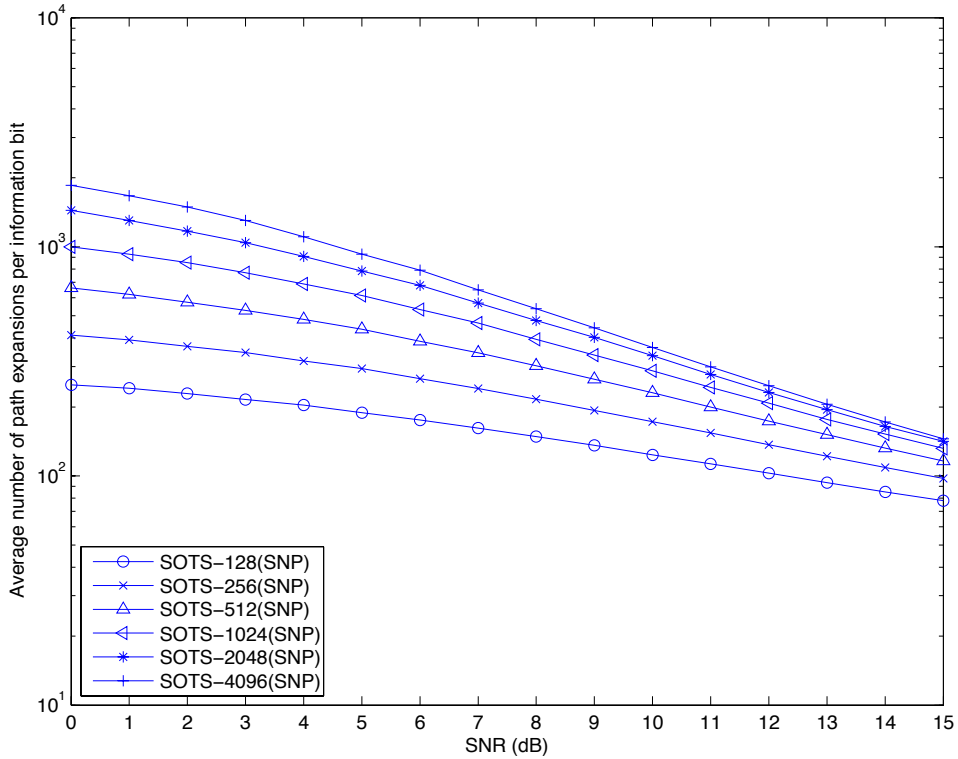


Figure 16: The average numbers of path expansions per information bit for for the self-orthogonal trellis-structure codes of length $N = 24$ with maximal state number 128, 256, 512, 1024, 2048 and 4096, respectively. The decoders used are the sequential decoding with no heuristic prediction (SNP). The code rate is 1/2.

Simulation Results

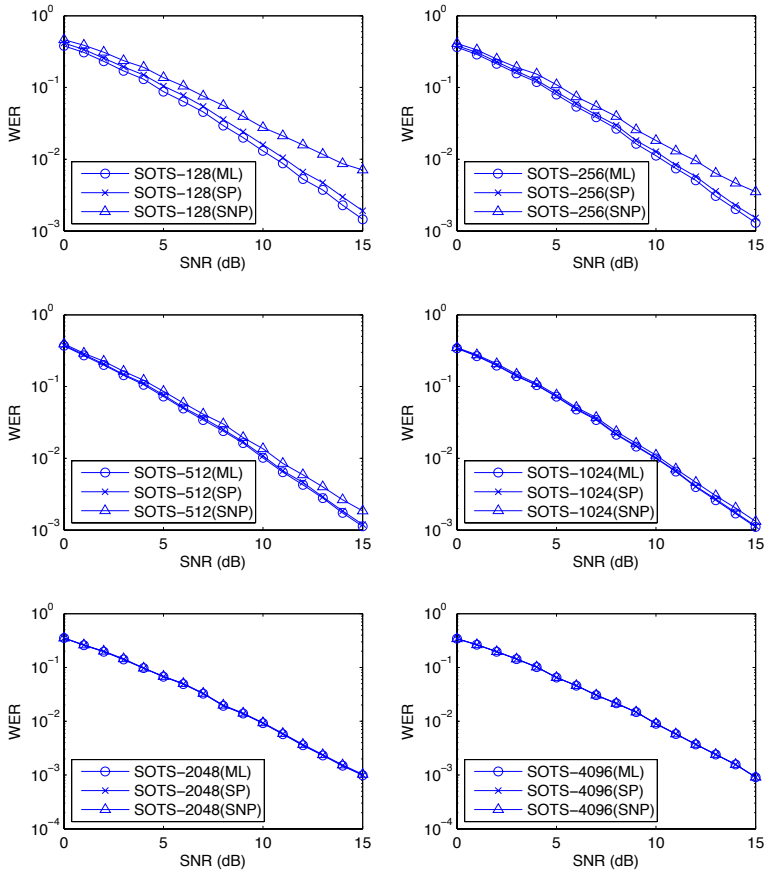


Figure 17: Word error rates (WERs) for the proposed codes respectively with $S = 128$, 256, 512, 1024, 2048 and 4096, decoded by different approaches: ML, SP and SNP. The code rate is 1/2 and the codeword length is $N = 24$.

Simulation Results

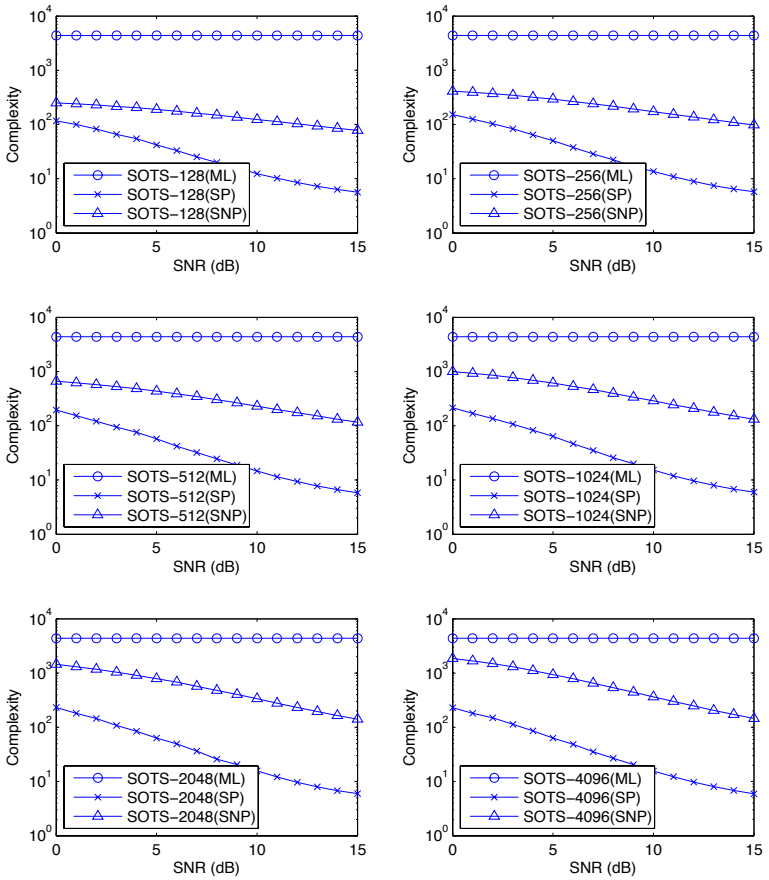


Figure 18: The average numbers of path expansions per information bit for the proposed codes respectively with $S = 128$, 256, 512, 1024, 2048 and 4096, decoded by different approaches: ML, SP and SNP. The code rate is $1/2$ and the codeword length is $N = 24$.

Future Work

- The complexity of compression grows exponentially when the code-word length increases. How to find a good theory for self-orthogonal trellis-structure codes so that they can be constructed directly (without using a tree-structure code) is another subject for future research.

Thank You for Your Listening