Optimizing the Free Distance of Error-Correcting Variable Length Codes

Amadou Diallo[1], *Claudio Weidmann*[2] *and Michel Kieffer*[1,3]

[1] L2S - CNRS - SUPELEC - Univ Paris-Sud
3 rue Joliot-Curie, 91192 Gif-sur-Yvette cedex, France
email : *diallo@lss.supelec.fr*

[2] INTHFT, Vienna University of Technology
1040 Vienna, Austria.
email : *claudio.weidmann@ieee.org*

[3] On sabbatical leave at LTCI - CNRS - Telecom ParisTech
75013 Paris, France.
email : *kieffer@lss.supelec.fr*

This paper considers the optimization of Error-Correcting Variable-Length Codes (EC-VLC), which are a class of joint-source channel codes. The aim is to find a prefix-free codebook with the largest possible free distance for a given set of codeword lengths, $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_M)$. The proposed approach consists in ordering all possible codebooks associated to $\boldsymbol{\ell}$ on a tree, and then to apply an efficient branch-and-prune algorithm to find a codebook with maximal free distance. Three methods for building the tree of codebooks are presented and their efficiency is compared.

Introduction

• **Goal**
– Design good error-correcting variable length codes (EC-VLC) regarding the free distance.

• **Approach**
– From a Kraft vector $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_M)$ such that $\sum_{i=1}^{M} 2^{-\ell_i} \leq 1$.
– Order in a tree all prefix codebooks associated to $\boldsymbol{\ell}$.
– Apply a *branch-and-prune* algorithm to find a codebook(s) with the largest free distance.

• **Difficulties**
– Number of corresponding codebooks can be very large.

Error-Correcting Variable Length Code

Basic definitions

– X a source with :
  – Alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$
  – Set of probabilities $\mathcal{P} = \{p_1, p_2, \ldots, p_M\}$
– EC-VLC associated to X is a codebook *i.e.*
  – A set of codewords $\mathcal{C} = \{c_1, c_2, \ldots, c_M\}$.
  – Associated to $\boldsymbol{\ell} = \{\ell_1, \ell_2, \ldots, \ell_M\}$.
  – c_i has length ℓ_i
– Entropy : $H = -\sum_{i=1}^{M} p_i \log_2 p_i$
– Average coding length : $\ell_{\text{av}} = \sum_{i=1}^{M} p_i \ell_i$
– A sequence $\boldsymbol{t} = c_i \circ c_j \circ \cdots \circ c_k$
– EC-VLC is characterized by its :
  1. Redundancy $R_{\text{c}} = \ell_{\text{av}} - H$
  2. Free distance

$$d_{\text{free}} = \min_{(\boldsymbol{t}_1, \boldsymbol{t}_2) \,:\, \boldsymbol{t}_1 \neq \boldsymbol{t}_2 \text{ and } \ell(\boldsymbol{t}_1) = \ell(\boldsymbol{t}_2)} d_{\text{H}}(\boldsymbol{t}_1, \boldsymbol{t}_2)$$

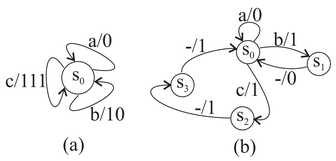
.

Computing the free distance of an EC-VLC

– EC-VLC : directed graph $\Gamma(\mathcal{S}, \mathcal{T})$
  – $\mathcal{S}$: Set of states
  – $\mathcal{T}$: Set of transitions (input/output labels)

***Example* 1** *: X_3, with $\mathcal{A}_3 = \{a_1, a_2, a_3\}$ and $\mathcal{C}_3 = \{0, 1, 111\}$*

– For computing d_{free}
  – derive a product graph
  – apply Dijkstra's algorithm



Graph of X_3 (a) and its corresponding bit clock graph (b)

Incomplete and complete EC-VLC

– EC-VLC is an *incomplete codebook* (IC) if some codewords or parts of codewords are not determined. For X_3, $\mathcal{C}^1 = \{0, 10, xxx\}$ and $\mathcal{C}^1 = \{0, 1x, 1xx\}$, x is an undetermined bit.

– A *complete codebook* (CC) is an EC-VLC in which all codewords are determined.

***Definition* 1** *: An (incomplete) codebook $\mathcal{C}^1$ is derived from an IC $\mathcal{C}^0$ (denoted $\mathcal{C}^0 \subset \mathcal{C}^1$) if it is obtained by specifying some (all) undetermined bits in $\mathcal{C}^0$.*

***Lemma* 1** *: If $\mathcal{C}^0 \subset \mathcal{C}^1$, then upper and lower bounds on d_{free} of $\mathcal{C}^1$ can be obtained from $\mathcal{C}^0$ lem :lemma1)*

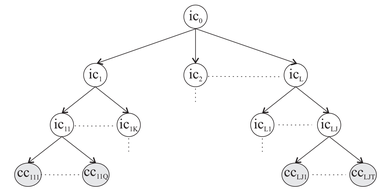
Approach

• Order in a tree all EC-VLC associated to $\boldsymbol{\ell}$

• Internal nodes : ICs

• leaves : CCs (EC-VLCs)

• Lemma1 in a *branch-and-prune* algorihtm

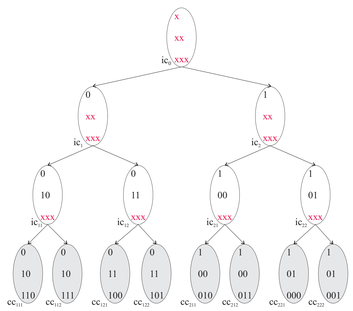


Tree of EC-VLC

Structuring the search space

Tree methods for building the tree of EC-VLC are proposed :

Construction by codeword



Tree of EC-VLCs : $\boldsymbol{\ell} = (1, 2, 3)$. By codewords

– Put an initial IC with all bits undetermined

– Specify one codeword at time

– Symmetry by inverting all bit for a CC

– Only half of the tree is needed for computing $d_{\text{free}}^{\max}$

– Lexicographic order : reduces number of EC-VLCs to explore

Construction by bit plane

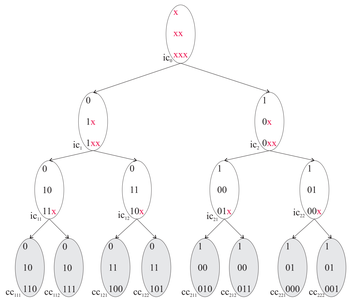
– Put an initial IC with all bits undetermined

– Specify the first bit of all codewords

– Then, specify the second bit, and so on

– At each step, verify the prefix condition

– The same symmetry and the same lexicographic order as the construction by codewords

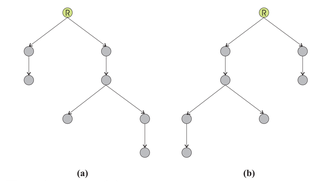


Tree of EC-VLCs : $\boldsymbol{\ell} = (1, 2, 3)$. By bit plane

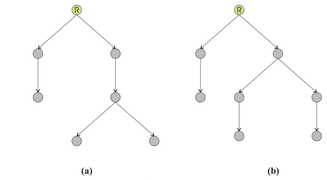
Construction by canonical tree

– EC-VLC can be represented with a binary tree
– The leaves represent codewords.
– Bounds of d_{free} can be obtained directly from the structure of the tree.
– To exploit this fact, group code trees into isomorphism (equivalence) classes.

***Definition* 2** *: Two binary trees are said to be isomorphic if they can be transformed into each other by transposing (flipping) the children of internal nodes, including the root* (i.e. *all nodes stay at the same level, only they horizontal changes, assuming the tree is drawn top-down from the root)*



Two isomorphic trees



two non-isomorphic trees

– Represent an isomorphism class by a *canonical tree*

***Definition* 3** *: T is canonical tree if it satisfies $left(T_i) \prec right(T_i)$ at all its internal nodes T_i* (i.e. *recursively from the root down).*

– All possible bits assignment for a canonical tree give all corresponding EC-VLCs.
– For code search for a given $\boldsymbol{\ell}$, list all canonical trees.
– Explore first the trees with the maximum bounds of d_{free}.
– The advantage of this method is that, a canonical tree groups large number of EC-VLC

Experimental results

– Display time saving over an exhaustive search gained by applying the branch-and-prune algorithm.

– Kraft vector for this experiment $\boldsymbol{\ell} = (3, 4, 5, 6)$.
– #VLCs is the number of intermediate EC-VLCs that were examined by the algorithm.
– For this simulation, $d_{\text{free}}^{\max} = 4$.

Methods	Exhaustive	Bitplane	Canon. tree	Codewors
#VLCs	72800	4070	3286	1222
Time [s]	2477	222	118	16

TABLE1: Comparison between exhaustive search and tree branch-and-prune algorithm

Conclusion

– Branch-and-prune algorithm fast method for designing good EC-VLCs.
– By codewords, better method for $\boldsymbol{\ell}$ with distinct lengths.
– By bit plane and canonical tree, better when many codeword lengths are equal.
– Future works : make simulation using the 26 English alphabet symbols.

[1] V. Buttigieg, "Variable length error-correcting codes," PhD dissertation, University of Manchester, Univ. Manchester, U.K., 1995
[2] A. Diallo, C. Weidmann, M. Kieffer, "Optimizing the search of finite-state joint source-channel codes based on arithmetic coding". 2009, Proc. Eusipco 2009, Glasgow
[3] A.Diallo, C. Weidmann, M. Kieffer, "Efficient computation an optimization of the free distance of variable-length finite-state joint source-channel codes," Dec. 2009, submitted to IEEE Trans. Commun