Optimised constructions for variable-length error correcting codes

Catherine Lamy[1], Johann Paccaut
Philips Recherche France, Suresnes, France
lamy@ieee.org

Abstract — **An optimised construction of variable-length error-correcting codes (VLEC) is proposed in this paper. Compared to the state-of-the-art, the three main improvements consist first in improving the codewords search algorithm complexity/efficiency trade-off, second in loosening the codewords deletion rule, and third in taking advantage of eventual previous searches. It is shown that the optimised algorithm can find good VLEC codes for alphabets up to 200 symbols, and outperforms existing algorithms.**

Index Terms — **variable-length error-correcting codes, joint source-channel coding, variable-length codes, error correction coding, code construction**

I. Introduction and motivation

Classically used in source coding for their compression capabilities, variable-length codes (VLC) [1] are often associated with channel coding techniques [2] which combat the effects of a transmission channel (fading, noise, ...). Implementing these two operations separately from each other is a direct consequence of Shannon's well known *separation* theorem [3], which states that the two operations can be done separately without asymptotical performance loss. However, this theorem neither holds for some classes of sources and/or channels, nor offers any guarantee in terms of complexity and practical feasibility. Since source coding aims at removing redundancy and channel coding aims at reintroducing it, it is also investigated how to efficiently co-ordinate the two techniques to improve the overall system while keeping an acceptable level of complexity [4]. Among the joint coding proposed solutions, one finds the variable-length error correcting (VLEC) codes [5, 6], which offer compression capability while providing error correction.

The VLEC codes ensure a minimal free distance over block-like sequences, which makes especially sense when used with the recently introduced approaches [7, 8, 9] for VLC decoding. These techniques have in common the fact that they consider the overall sequence of variable-length codewords to perform Maximum A Posteriori (MAP) decoding, rather than decoding each codeword instantaneously. Working whether on trellises or code trees, these algorithms complement at the decoding part the VLEC joint source and channel coding technique by taking advantage of the VLEC code free distance.

Introduced in 1995, the promising "Heuristic" construction algorithm for VLEC codes proposed in [6] for short alphabets remains very time consuming, becoming even prohibitive for higher length alphabets sources. In this paper, we explore several optimisation techniques that significantly decrease the complexity of the Heuristic construction, and consequently allow to find good VLEC codes for large alphabets sources.

[1]C. Lamy is now with THALES Communications France, Gennevilliers, France (catherine.lamy@fr.thalesgroup.com).

The paper is organised as follows. After the introduction of some notations and definitions, the state-of-the-art Heuristic algorithm is reviewed in Section II. The proposed optimisation techniques are presented in Section III and numerical results on their application are provided in Section IV. Finally, some conclusions are drawn.

II. State-of-the-art

A. Definitions and notations

Let $\mathcal{C}$ be a uniquely decodable [10] variable-length code of cardinality $\mathcal{N}_{\mathcal{C}}$. Let $|\mathbf{x}^i|$ and $P(\mathbf{x}^i)$ denote the length and the probability of occurence of the data source symbol mapped into word $\mathbf{x}^i = (x^i_1, .., x^i_{|\mathbf{x}^i|})$ in $\mathcal{C}$, where $\sum_{i=1}^{|\mathcal{C}|} P(\mathbf{x}^i) = 1$. Let σ be the number of different word lengths in $\mathcal{C}$. We denote these different lengths by $L_1, L_2, \ldots, L_\sigma$, where $L_1 < L_2 < \ldots < L_\sigma$, and the corresponding number of codewords by $n_1, n_2, \ldots, n_\sigma$, where $\sum_{i=1}^{\sigma} n_i = \mathcal{N}_{\mathcal{C}}$.

Definition 1 *The* average length $AL = \sum_{i=1}^{|\mathcal{C}|} |\mathbf{x}^i| P(\mathbf{x}^i)$ *of a code $\mathcal{C}$ is the average number of bits needed to transmit a word.*

Definition 2 *Let $\mathbf{f}_i = \mathbf{x}^{i_1}\mathbf{x}^{i_2}\cdots\mathbf{x}^{i_n}$ be a concatenation of n words of $\mathcal{C}$. The set $F_N = \{f_i : |f_i| = N\}$ is called the* extended code *of $\mathcal{C}$ of order N.*

Definition 3 *The* Hamming weight $w(\mathbf{x})$ *of a word* $\mathbf{x}$ *is the number of non-zero symbols in* $\mathbf{x}$. *The* Hamming distance $h(\mathbf{x}^i, \mathbf{x}^j)$ *between two words $\mathbf{x}^i$ and $\mathbf{x}^j$ of equal length is the number of positions in which $\mathbf{x}^i$ and $\mathbf{x}^j$ differ.*

Definition 4 *The* minimum block distance b_k *associated to the codeword length L_k of a code $\mathcal{C}$ is defined as the minimum Hamming distance between all distinct codewords of $\mathcal{C}$ with length L_k.*

$$b_k = \min_{\substack{(\mathbf{x}^i,\, \mathbf{x}^j) \in \mathcal{C},\, i \neq j \\ |\mathbf{x}^i| = |\mathbf{x}^j| = L_k}} \left\{ h(\mathbf{x}^i, \mathbf{x}^j) \right\}. \quad (1)$$

The overall minimum block distance $b_{\min}$ *of a code $\mathcal{C}$ is the minimum block distance value for every possible length L_k: $b_{\min} = \min_{1 \le k \le \sigma} b_k$.*

Definition 5 *The* diverging distance $d(\mathbf{x}^i, \mathbf{x}^j)$ *(resp. the* converging distance $c(\mathbf{x}^i, \mathbf{x}^j)$*) between two codewords of different lengths $|\mathbf{x}^i|$ and $|\mathbf{x}^j|$ of a code $\mathcal{C}$ is defined as the Hamming distance between the ℓ-length prefixes (resp. suffixes) of codewords $\mathbf{x}^i$ and $\mathbf{x}^j$, with* $\ell = \min\{|\mathbf{x}^i|, |\mathbf{x}^j|\}$.

$$d(\mathbf{x}^i, \mathbf{x}^j) = h(x^i_1 \cdots x^i_\ell, x^j_1 \cdots x^j_\ell), \quad (2)$$

$$c(\mathbf{x}^i, \mathbf{x}^j) = h(x^i_{|\mathbf{x}^i|-\ell+1} \cdots x^i_{|\mathbf{x}^i|}, x^j_{|\mathbf{x}^j|-\ell+1} \cdots x^j_{|\mathbf{x}^j|}). \quad (3)$$

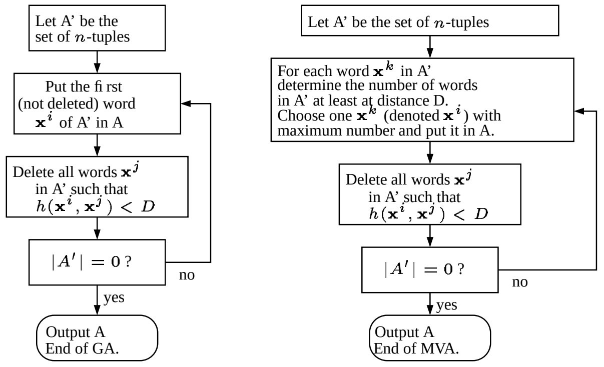
Let A' be the set of n-tuples
Put the first (not deleted) word $\mathbf{x}^i$ of A' in A
Delete all words $\mathbf{x}^j$ in A' such that $h(\mathbf{x}^i, \mathbf{x}^j) < D$
$|A'| = 0$?
no
yes
Output A End of GA.
Let A' be the set of n-tuples
For each word $\mathbf{x}^k$ in A' determine the number of words in A' at least at distance D. Choose one $\mathbf{x}^k$ (denoted $\mathbf{x}^i$) with maximum number and put it in A.
Delete all words $\mathbf{x}^j$ in A' such that $h(\mathbf{x}^i, \mathbf{x}^j) < D$
$|A'| = 0$?
no
yes
Output A End of MVA.

Figure 1: Flowcharts of the GA and MVA algorithms.

The minimum diverging distance $d_{\min}$ *(resp.* minimum converging distance $c_{\min}$*) of a code $\mathcal{C}$ is the minimum value for all diverging (resp. converging) distances between every possible couple of codewords in $\mathcal{C}$:* $d_{\min} = \min_{(\mathbf{x}^i,\mathbf{x}^j)\in\mathcal{C}, |\mathbf{x}^i|\neq|\mathbf{x}^j|} \left\{d(\mathbf{x}^i,\mathbf{x}^j)\right\}$ *and* $c_{\min} = \min_{(\mathbf{x}^i,\mathbf{x}^j)\in\mathcal{C}, |\mathbf{x}^i|\neq|\mathbf{x}^j|} \left\{c(\mathbf{x}^i,\mathbf{x}^j)\right\}$.

Definition 6 *The* free distance d_{free} *is the minimum Hamming distance in the set of all arbitrary extended codes:* $d_{free} = \min\left\{h(\mathbf{f}^i, \mathbf{f}^j) : \mathbf{f}^i, \mathbf{f}^j \in F_N, N = 1, \ldots, \infty\right\}$.
The free distance d_{free} of a VLEC code is bounded by ([6]):

$$d_{free} \geq \min\left(b_{\min}, d_{\min} + c_{\min}\right) \tag{4}$$

Variable-length error correcting (VLEC) codes are defined [6] as codes ensuring a free distance over block-like sequences, and are denoted using the following structure: $(n_1@L_1, b_1; n_2@L_2, b_2; \ldots; n_\sigma@L_\sigma, b_\sigma; d_{\min}, c_{\min})$.

B. Heuristic construction [6]

The Heuristic method [5, 6] relies on computer search to construct an $\mathcal{N}_\mathcal{C}$ symbol VLEC code $\mathcal{C}$ with specified overall minimum block, diverging and converging distances and with codeword lengths matched to the source statistics so as to obtain minimal average length for the chosen free distance. In practice, one takes $b_{\min} = d_{\min} + c_{\min} = d_{free}$ and $d_{\min} = \lceil d_{free}/2 \rceil$. The Heuristic algorithm main steps are:
- $\mathcal{C}$ is initially set equal to a fixed-length code of length L_1 and minimal distance $b_{\min}$. This step, as all following determinations of sets of same length words at a given distance, is carried out either by the greedy algorithm (GA) or the majority voting algorithm (MVA) [6]. These algorithms, recalled in Fig. 1, generate sets of n-bits long words distant of D.
- a set W is created, that contains all L_1-tuples with distance $d_{\min}$ of each codeword in $\mathcal{C}$. If W is not empty, one extra bit is affixed at the end of all its words. This new set with twice more words replaces the previous W. Each word in W has length equal to $L_1 + 1$.
- all words in W that do not satisfy the $c_{\min}$ distance with all codewords of $\mathcal{C}$ are deleted. At this point, W satisfies both $d_{\min}$ and $c_{\min}$ distances with the VLEC code $\mathcal{C}$.
- it is checked that all words of the set W are distant of $b_{\min}$ with help of the GA or the MVA applied to the set W. Kept words are then added to code $\mathcal{C}$.

The algorithm is repeated until it finds the targeted number n of codewords or has no more possibility to continue. In the case where the number of codewords in $\mathcal{C}$ is larger than or equal to n, the average length of the corresponding candidate structure is calculated and compared to the current best one. If it is better, this new AL value and the corresponding VLEC are stored. The algorithm goes on with deleting codewords among the last added group: half the group for the GA and the "best" one for the MVA [6], and continues with above described algorithm steps. In practice, the GA is quicker but non-optimal, whereas the MVA is slower but more efficient. In the case where no word is found, or if the maximal length is reached, shorter length codewords are deleted, following the deletion pattern mentioned above. Such deletions allow to obtain more longer codewords, and are repeated until one finds a possible VLEC structure.

III. Optimising the VLEC code construction

Considering the Heuristic algorithm described above, we first suggest a trade-off between the GA and the MVA. Then, we propose to gain in simplicity by avoiding very unlikely structures, by loosening the codewords deletion rule, and finally by taking advantage of eventual previous searches.

A. Greedy algorithm by step (GAS)

Following the idea that there should be a good compromise on the use of the GA and MVA search algorithms and their corresponding deletion rules, we propose a slightly modified algorithm. Called GAS for *Greedy Algorithm by Step*, this solution relies on the GA search method, but only the last codeword of the considered word group is deleted during the deletion phases[1]. Searching over more solutions, the GAS is consequently slower than the GA but offers similar or better average length, as shown by simulation.

B. noHole optimisation

Based on the observation that whether in our simulations or in litterature, almost none of the obtained best codes has a hole (or jump length) in its structure length, *i.e.* no codeword at a given length, we propose to consider that most good codes do not have jump of length and reduce accordingly the set of examined VLEC codes. Following this hypothesis, the Heuristic algorithm is modified, and useless parts are removed, leading to complexity reduction.

C. L_s optimisation

Considering the evolution of the codewords stack in the classical Heuristic process, we propose to perform the codeword deletion not only in the last obtained codewords group, but more generally in any given length value group. This way, it is possible to go back directly to smaller lengths, *i.e.* to skip many algorithm steps in cases where there are too many small length codewords. Denoting by L_s (with s for skip) the length to which the algorithm will skip back in the codeword deletion stage, we propose to skip parts of the original algorithm by carefully jumping to lower lengths when looking for codewords to be deleted. Naturally, when the considered codewords group length L is smaller than L_s, the classical method is applied. Lengths between L_1 and L_s are called *free lengths*,

[1]As with the MVA, this may cause the loss of the resulting code linearity.

i.e. lengths with a freedom degree, which are decremented one by one in the search process. Of course, when the number of free lengths grows up, so does the simulation time. In practice, simulation results show that very good compression rates can be obtained for $L_s < L_{\max}$, where $L_{\max}$ is the maximal authorised codeword length. We observed that increasing the value of L_s resulted in an improvement in AL up until a constant floor (best value). This behaviour prompts for a dynamic choice of L_s, starting with $L_s = L_1$, and incrementing it until the floor is reached.

D. BestAllure optimisation

This last optimisation, that we have called BestAllure, proceeds from the remark that when updating the L_s parameter for a new search, no advantage is taken from eventual previous searches. Following this lead, we propose to establish a semi-recursive way to reach quickly higher L_s values, allowing us to find better compression gains for acceptable computation time. In practice, we will use this last optimisation with the previously introduced GAS, noHole and Ls ones (eventhough it is not a necessity).

We propose to keep in memory the beginning of the best VLEC structure of each L_s, and to re-use this knowledge within the search with the next $L'_s = L_s + 1$ value. As L_s rises, the size of the beginning stored in memory increases accordingly to avoid a resulting increase of the free length, that would exponentially impact on the computation time. In fact, simulations tend to show that when L_s increases, the beginning of each code remains (quasi) constant for always more lengths, justifying our interest in the pre-computed information. Whereas the algorithm previously tested all combinations for lengths in $[L_1, L_s]$, it now does it only in a reduced interval $[L_{k+1}, L_s]$, where L_k represents the higher length of the code beginning stored in memory. The number of free lengths is now $N_{fl} = L_s - L_k$. To offer more flexibility to the algorithm, we have even defined three levels of freedom:
- *fixed length*, which corresponds to the lengths where the number of codewords is fixed from previous stages,
- *variance length*, which is the set of lengths where some freedom (*e.g.* ± 1 around the fixed codeword number) is allowed,
- *free length*, which corresponds to the set of lengths where all possible numbers of codewords are tested.

As with L_s optimisation, the rest of the length distribution, or *tail lengths*, corresponds to lengths above the L_s limit.

The variance length part results directly from practical observations, when it appeared during simulations that to improve the results it was necessary to establish a slight freedom on the number of codewords found at the higher fixed length L_k. Naturally, a generalisation of this variance length set can also be considered for length smaller than L_k.

Together with the previous ones, the BestAllure optimisation provides huge gains of time. In practice, they are especially interesting for a source with high number of symbols, where exhaustive methods cannot be applied and where existing ones, like Buttigieg's one [6] are untractable.

IV. NUMERICAL RESULTS

To illustrate the different optimisations introduced in this paper, we consider two sources: the first one is the well-known 26-symbol English source [6] and the second one is a 208-symbol MPEG-4 source, whose statistics are given in Table 1. All simulations were carried out on the same Sun Ultra-80 computer, 440 MHz with full-time cpu, to allow the comparison of the algorithms execution time.

symbol probability	number of symbols
0.125254	2
0.062622	2
0.031311	6
0.015656	6
0.007828	20
0.003914	17
0.001957	27
0.000978	44
0.000489	28
0.000245	24
0.000122	32

Table 1: MPEG-4-like source statistics.

	$d_{free} = 3$		$d_{free} = 5$		$d_{free} = 7$	
method	AL	time	AL	time	AL	time
GA	6.4946	4 s	8.5061	15 s	10.79	2 min
MVA	6.3038	50 s	8.4752	16 min	10.7385	14 h
GAS	6.3494	42 s	8.5061	3 min	10.79	13 min
noHole	6.3530	17 s	8.5061	45 s	10.79	3 min

Table 2: VLEC codes for the 26-symbol English source with GA, MVA, GAS and GAS+noHole optimisation.

with GA

L_1	AL	time
3	9.0448	24 h
4	8.0464	41 h
5	7.9822	48 h
6	8.1106	20.5 h
7	8.3645	3 h
8	8.8054	25 min
9	9.3118	4 min

with MVA+L_s optimisation

L_1	L_s	AL	time
5	9	8.0331	25 min
	10	7.9527	38 h
9	9	9.3	60 s
	10	9.285	2.5 h

GA+L_s optimisation for $L_1 = 5$

L_s	AL	time
8	8.3133	2.5 s
9	8.0352	15 s
10	8.0086	92 s
11	7.9953	580 s
12	7.9822	80 min
13	7.9822	8 h
14	7.9822	48 h
15	7.9822	48 h

Table 3: VLEC codes for the 208-symbol MPEG-4 source with $d_{free} = 3$ and $L_{\max} = 15$.

Reference results obtained for the English source with the Heuristic algorithm for the GA and the MVA are given in Table 2 for different values of d_{free}. As foreseen, the MVA searches take much more time than the GA ones, and the GAS modification offers same or better average length for higher search time than the GA. Combining the GAS with the noHole optimisation, it appears that noHole provides a time reduction factor of about 3 when compared to GAS, with only a slight AL degradation. As for the MVA, the GAS complexity, with

	$d_{free}=3$ L_1=4 $L_{\max}$=13		$d_{free}=5$ L_1=6 $L_{\max}$=15		$d_{free}=7$ L_1=8 $L_{\max}$=15	
L_s	AL	time (s)	AL	time (s)	AL	time (s)
6	6.6356	0.017	-	-	x	x
7	6.6350	0.08	8.5061	0.01	x	x
8	6.6350	0.5	8.5061	0.03	10.79	0.02
10	6.6350	6.7	8.5061	0.76	10.79	0.17
13	6.6350	17	8.5061	45	10.79	12.8
15	x	x	x	x	10.79	175

Table 4: Influence of noHole+L_s optimisations for the 26 symbol English source.

	$L_1=5$		$L_1=6$		$L_1=7$	
L_s	AL	time	AL	time	AL	time
8	8.3133	8 s	8.4905	13 s	8.5643	5 s
9	8.0346	8 min	8.1186	8 min	8.3413	3 min
10	7.9586	5 h	8.0737	4 h	8.3055	2 h
11	7.9392	1 week	.	.	8.2937	134 h

Table 5: Influence of L_s optimisation for the 208 symbol MPEG-4 source with GAS+noHole method.

or without noHole optimisation, would however be prohibitive when applied to the MPEG-4 source. Table 3 shows as a reference the codes obtained with GA algorithm for the MPEG-4 source for various values of L_1. Focussing on minimal length $L_1 = 5$ which provides the best code with the GA, we have applied noHole and L_s optimisations for various L_s values and found that it was possible to obtain a noticeable reduction factor, going from 48 hours to merely 80 minutes computation time. For the same parameters, it was possible to combine the MVA and the noHole and L_s optimisations, reaching even an better AL.

Results obtained with L_s optimisation for the English source are presented in Table 4, where crosses (x) indicate impossible values for L_s and where dashes $(-)$ to cases where the algorithm could not find any solution for the given initial parameters. Comparing these results with those in Table 2, it appears possible to obtain better or at least equivalent results to GA algorithm with a time reduction factor larger than 10. The time computation gain offered by this L_s optimisation affords us to find computation gain from about 5 for $d_{free} = 3$ to about 2500 for $d_{free} = 7$.

The gain offered by L_s optimisation is even more noticeable with the MPEG-4 source. Indeed, the results given in Table 5 justify the use of the GAS together with the noHole and L_s optimisations: while the best AL achieved with GA and L_s optimisation leads to $AL = 7.9822$ for two days computation time, we find with GAS a better VLEC code with $AL = 7.9586$ in only 5 hours. Moreover, it is to be noted that this result is extremely close to the best one obtained with the MVA algorithm combined with noHole and L_s optimisations (reaching $AL = 7.9527$ in 38 hours, as shown in Table 3), and that GAS provides an even better code with $AL = 7.9392$ with $L_s = 11$ in one week. With such a parameter the MVA complexity is prohibitive.

Finally, Table 6 shows the results obtained for the MPEG-4 source with BestAllure optimisation for $N_{fl} = 3$ or 4

	with N_{fl}=3		with N_{fl}=4	
L_s	AL	time	AL	time
8	8.0162	27 s	x	x
9	7.9734	7 min	8.0346	6 min
10	7.9489	14 min	7.9586	3 h
11	7.9475	32 min	7.9392	8 h
12	7.9464	1 h	7.9379	21 h
13	7.9455	2.5 h	7.9379	65 h

Table 6: Results of BestAllure optimisation for the 208 symbol MPEG-4 source.

free lengths. The best overall VLEC code for this source ($AL = 7.9379$ for code (1@5,-;2@6,3;7@7,3;9@8,3;8@9,3;12@10,3;21@11,3;30@12,3;37@13,3;40@14,3;41@15,3;2,1)) is found in about 32 hours. This confirms that BestAllure allows for huge gain of time while providing very good results.

V. Conclusions

We have presented in this paper various optimisations for the Heuristic construction method of variable-length error correcting codes introduced in [6]. It was in particular shown that better VLEC codes could be obtained by taking advantage from previous searches in the BestAllure semi-recursive optimisation. Especially interesting for sources with a large number of symbols, where exhaustive methods are prohibitive, our optimisations allow to obtain codes fitted to the considered source while keeping error-correction capability.

Acknowledgments

The authors would like to thank Lisa Perros-Meilhac for fruitful discussions on the subject.

References

[1] D.A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," in *Proceedings of the Institute of Radio Engineers*, vol. 40, pp. 1098–1101, September 1952.

[2] S. Lin and D.J. Costello, *Error control coding: fundamentals and applications*. Englewoods Cliffs, New Jersey, USA: Prentice Hall, 1983.

[3] C.E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pt. I, pp. 379–423, 1948; pt. II, pp. 623–656, 1948.

[4] J.L. Massey, "Joint source and channel coding", *in Communication Systems and Random Process Theory, NATO Advanced Studies Institutes Series E25*, pp. 279-293. Sijthoff & Noordhoff, Alphen aan den Rijn, The Netherlands, 1978.

[5] V. Buttigieg and P.G. Farrell, "Constructions for variable-length error-correcting codes", in *Proceedings of Cryptography and Coding, 5th IMA Conference*, Cirencester, UK, pp. 282–291, December 1995.

[6] V. Buttigieg, *Variable-length error-correcting codes*. Ph.D. dissertation, University of Manchester, Manchester, United Kingdom, 1995.

[7] N. Demir and K. Sayood, "Joint source/channel coding for variable length codes," in *Proceedings of the Data Compression Conference*, pp. 139–148, Snowbird, USA, March-April 1998.

[8] M. Park and D. J. Miller,"Joint source-channel decoding for variable-length encoded data by exact and approximate MAP sequence estimation," *IEEE Transactions on Communications*, vol. 48(1), pp. 1–6, January 2000.

[9] L. Perros-Meilhac and C. Lamy, "Huffman tree based metric derivation for a low-complexity sequential soft VLC decoding," in *Proceedings of ICC'02*, pp. 783–787, New York, USA, 2002.

[10] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, chap. 5, New York: John Wiley & Sons, 1991.