Optimal Single- and Multiple-Tree Almost Instantaneous Variable-to-Fixed Codes

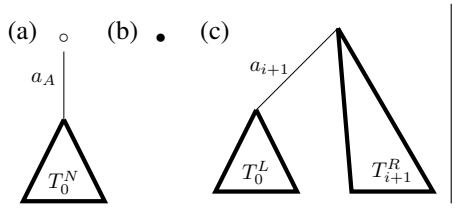
Danny Dubé Fatma Haddad
Université Laval, Quebec City, Canada
Email: Danny.Dube@ift.ulaval.ca Fatma.Haddad.1@ulaval.ca

Abstract

Variable-to-fixed codes are often based on dictionaries that obey the prefix-free property. In particular, the Tunstall algorithm builds such codes [1]. However, the prefix-free property is not necessary to have correct variable-to-fixed codes. Removing the constraint to obey the prefix-free property may offer the opportunity to build more efficient codes [2], [3]. Here, we come back on the almost instantaneous variable-to-fixed (AIVF) codes introduced by Yamamoto and Yokoo. They considered both single trees and multiple trees to perform the parsing of the source data. We show that, in some cases, their techniques build suboptimal codes. We propose potential correctives to their techniques. We also propose a new, completely different technique based on dynamic programming that builds optimal dictionary trees.

We identified two defects in the Yamamoto-Yokoo technique (YY) that prevents it to build optimal AIVF codes. In multiple-tree mode, YY initializes the dictionary tree with a complete root. While a complete root is a necessity in single-tree mode, to ensure progress during parsing, it is *not* in multiple-tree mode. The obligation to complete the root in multiple-tree mode sometimes leads to the construction of suboptimal codes. In both modes, YY works by having two strategies compete to grow dictionary trees: one consists in completing the most probable incomplete node and the other one consists in growing a few best new leaves. When the proposition made by the second strategy happens to be the most competitive, it is harmful to fully adopt the proposition. We proposed two correctives to compensate for the identified defects.

In order to build clearly optimal dictionary trees, we proposed a simple construction technique based on dynamic programming (DP) [4], [5]. The technique builds all trees T_i^N up to a desired size, where N is the size of the tree in terms of number of parsewords and i is the strength of the partial information about the next symbol of the input. Here are the fundamental shapes of the dictionary trees built by the DP algorithm, where the alphabet is $\{a_1, \ldots, a_A, \}$ such that $p(a_j) \geq p(a_{j+1})$.



(a) $T_{A-1}^N = \text{DEFAULT}(T_0^N)$
(b) $T_i^1 = \text{ROOT}$, where: $i \leq A-2$
(c) $T_i^N = T_0^L \oplus T_{i+1}^R$, where: $i \leq A-2$, $2 \leq N = L+R$

As future work, we should verify whether applying our correctives on YY would make it optimal. The "almost instantaneous" property remains a constraint on the considered VF codes, even if it is looser than the PF property, and we should investigate on the opportunities offered by the removal or relaxation of this constraint; e.g., codes with a longer delay [6].

REFERENCES

[1] B. P. Tunstall, *Synthesis of Noiseless Compression Codes*, Ph.D. thesis, Georgia Institute of Technology, 1967.

[2] S. A. Savari, "Variable-to-fixed length codes and plurally parsable dictionaries," in *Proc. of the Data Compression Conference*, Mar. 1999, pp. 453–462.

[3] H. Yamamoto and H. Yokoo, "Average-sense optimality and competitive optimality for almost instantaneous VF codes," *IEEE Transactions on Information Theory*, vol. 47, no. 6, pp. 2174–2184, Sep. 2001.

[4] S.-L. Chen and M. J. Golin, "A dynamic programming algorithm for constructing optimal "1"-ended binary prefix-free codes," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1637–1644, 2000.

[5] K. Iwata and H. Yamamoto, "A dynamic programming algorithm to construct optimal code trees of AIVF codes," in *Proc. of the International Symposium on Information Theory and Applications*, Nov. 2016.

[6] K. Iwata and H. Yamamoto, "An iterative algorithm to construct optimal binary AIFV-m codes," in *Proc. of the IEEE Information Theory Workshop*, Nov. 2017, pp. 519–523.