Analysis of a Block Arithmetic Coding: Discrete Divide and Conquer Recurrences

Michael Drmota
Inst. Discrete Mathematics and Geometry,
TU Wien,
A-1040 Wien, Austria,
Email: michael.drmota@tuwien.ac.at

Wojciech Szpankowski
Department of Computer Science,
Purdue University,
West Lafayette, IN 47907-2066 U.S.A.,
Email: spa@cs.purdue.edu

Abstract—In 1993 Boncelet introduced a block arithmetic scheme for entropy coding that combines advantages of stream arithmetic coding with algorithmic simplicity. It is a variable-to-fixed length encoding in which the source sequence is partitioned into variable length phrases that are encoded by a fixed length dictionary pointer. The parsing is accomplished through a complete parsing tree whose leaves represent phrases. This tree, in its suboptimal heuristic version, is constructed by a simple divide and conquer algorithm, whose analysis is the subject of this paper. For a memoryless source, we first derive the average redundancy and compare it to the (asymptotically) optimal Tunstall’s algorithm. Then we prove a central limit theorem for the phrase length. To establish these results, we apply powerful techniques such as Dirichlet series, Mellin-Perron formula, and (extended) Tauberian theorems of Wiener-Ikehara.

I. INTRODUCTION

We present a comprehensive analysis of a data compression algorithm due to Boncelet [3] known as Block Arithmetic Coding (BAC). Boncelet’s algorithm is a variable-to-fixed data compression scheme. To recall, a variable-to-fixed length encoder partitions a source string over an m-ary alphabet A into a concatenation of variable-length phrases. Each phrase belongs to a given dictionary of source strings. A uniquely parsable dictionary is represented by a complete parsing tree, i.e., a tree in which every internal node has all m children nodes. The dictionary entries correspond to the leaves of the associated parsing tree. The encoder represents each parsed string by a fixed length binary code corresponding to its dictionary entry. There are several well known variable-to-fixed algorithms; e.g., Tunstall and Khodak schemes (cf. [9], [16], [24]). Boncelet’s algorithm is based on a divide and conquer strategy, and therefore is fast and easy to implement.

Arithmetic entropy coders have been intensively studied in literature [8], [19], [20]. They are stream coders: an arbitrary long input sequence outputs a corresponding output stream. One disadvantage is that long input blocks are prone to the effect of transmission errors. Furthermore, in some applications the encoding and decoding are too complicated to be done in real time. On the other hand, Tunstall variable-to-fixed length scheme requires searching a codebook to find the most probable input sequence for the next splitting. To circumvent these difficulties, Boncelet designed a simple divide and conquer scheme that we briefly describe next.

In its simplest form, Boncelet builds a parsing tree by splitting a fixed number n of leaves (codewords) into subtrees of predetermined number of leaves. The number of leaves in each subtree is proportional to the probability of the alphabet symbols. For example, for a binary alphabet with probabilities p_1 and p_2 = 1 – p_1 the expected phrase length d(n) satisfies the following recurrence (other parameters such as variance, generating function of the phrase length fulfill similar recurrences)

\[ d(n) = 1 + p_1 d([pn + \delta]) + p_2 d([pn - \delta]) \]

where is \( \delta \) is a constant. This equation is an example of a general discrete (represented by the floor and ceiling functions) divide and conquer recurrence that we studied extensively in [10]. We shall adopt it here in order to present a comprehensive analysis of the Boncelet's algorithm performance including its redundancy and limiting distribution of the phrase length.

The question arises how the Boncelet algorithm compares to the (asymptotically) optimal Tunstall algorithm. In this work we provide an answer by first computing the redundancy of the Boncelet scheme (i.e., the excess of code length over the optimal code length) and compare it to the redundancy of the Tunstall code. Then we also establish that the phrase length of Boncelet’s scheme obeys the central limit law, as the Tunstall algorithm [9].

Literature on Boncelet’s algorithm and discrete divide and conquer recurrences is very scarce. To the best of our knowledge, there is no redundancy analysis for the Boncelet’s algorithm. In [3] some bounds on the average phrase length are derived. The Central Limit Law for the phrase length is new, too. Furthermore, we believe our contribution goes beyond analyzing precisely Boncelet’s algorithm performance. We accomplish it by developing a methodology for solving general discrete divide and conquer recurrences (cf. [10]). The literature on continuous divide and conquer recurrence is very extensive [1], [6], [5], however, the discrete version of the recurrence has received much less attention. Flajolet and Golin [12] and Cheung et al. [4] use similar techniques to ours, however, their recurrences are much simpler and restricted to \( p_1 = 0.5 \) (see also [11], [15]). We apply a combination of methods such as Tauberian theorems and Mellin-Perron techniques.
ACKNOWLEDGMENT

This work was supported in part by the Austrian Science Foundation FWF Grant No. S9604, by NSF STC Grant CCF-0939370, NSF Grants DMS-0800568, and CCF-0830140, and the MNSW grant N206 369739.

REFERENCES