Enumerative encoding/decoding of variable-to-fixed-length codes for memoryless sources

Yuriy A. Reznik
Corporate R & D
QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121
E-mail: yreznik@ieee.org

Anatoly V. Anisimov
Department of Cybernetics
Kiev National T. Schevchenko University
2 Acad. Glushkov Avenue, Bldg. 3
03680 Kiev, Ukraine
Email: ava@unicyb.kiev.ua

Abstract

We offer novel algorithms for efficient encoding/decoding of variable-to-fixed length codes, requiring at most quadratic amount of space: $O(L^2)$, where L is the depth of a coding tree. This is a major improvement compared to exponential $O(2^L)$ usage of space by conventional techniques using complete representations of coding trees in computer's memory.

These savings are achieved by utilizing algebraic properties of VF coding trees constructed by using Tunstall or Khodak algorithms, and employing combinatorial enumeration techniques for encoding/decoding of codewords. The encoding/decoding complexity of our algorithms is linear with the number of symbols they process.

As a side product, we also derive an *exact* formulae for the average redundancy of such codes under memoryless sources, and show its usefulness for analysis and design of codes with small number of codewords.

1 Definitions

Consider a memoryless source S producing symbols from an input alphabet $A = \{a_1, \ldots, a_m\}$ $(2 \leqslant m < \infty)$ with probabilities $\{p_i = P(a_i),\ i = 1, \ldots, m\}$. By $p_{\min}, p_{\max}$ we denote the probabilities of the least- and the most-likely symbols correspondingly, and assume that $0 < p_{\min} \leqslant p_{\max} < 1$.

Assume that a message $\Sigma = \{a_{i_k}\}_{k=1}^{\infty}$ is an infinite-length sequence of symbols produced by the source S. The main idea of variable length coding is to define a set of words $X = \{x_j \in A^*, j = 1, \ldots, M\}$ $(m \leqslant M < \infty)$, such that any message Σ can be uniquely represented by a sequence of words from X, and then map these words x_j into codewords $\phi(x_j)$ formed from letters of an output alphabet $B = \{b_1, \ldots, b_n\}$ $(2 \leqslant n < \infty)$:

$$\{a_{i_k}\}_{k=1}^{\infty} = \{x_{j_r}\}_{r=1}^{\infty} \to \{\phi(x_{j_r})\}_{r=1}^{\infty} = \{b_{i_s}\}_{s=1}^{\infty} .$$

We further assume that the mapping ϕ is injective, and that the output code $\{\phi(x_j)\}$ is uniquely decodable [4].

In this paper we only consider coding systems producing codewords $\phi(x_j)$ with the same length. For example, we can simply pick $|\phi(x_j)| = \lceil \log_n M \rceil$ $(1 \leqslant j \leqslant M)$, and use indices of words x_j to produce their codes. Such coding systems are called *variable-length-to-block* (VB) or *variable-to-fixed-length* (VF) codes.

The problem of construction of a VF code for a given source S consists in finding a prefix-free set X of constrained size $|X| \leqslant M$, such that the *average redundancy* of encoding of this source:

$$R_{\mathrm{VF}}(X,S) = \frac{\lceil \lg M \rceil}{d(X,S)} - h(S)\,. \tag{1}$$

is minimal. Here $d(X,S)$ denotes the *average delay* (or average length of a word in X):

$$d(X,S) = \sum_{j=1}^{M} P(x_j)\,|x_j|\ , \tag{2}$$

and $h(S)$ is the entropy of the source:

$$h(S) = -\sum_{i=1}^{m} p_i \lg p_i\,. \tag{3}$$

In some cases it might be sufficient to find prefix set X minimizing the *idealized average redundancy*:

$$R^{*}_{\mathrm{VF}}(X,S) = \frac{\lg M}{d(X,S)} - h(S)\,. \tag{4}$$

but these problems are very similar.

2 Tunstall and Khodak Algorithms

The best known algorithm for construction of optimal VF-codes for memoryless sources is one, due to B. P. Tunstall [21] (see also [6]). It is remarkably simple: start with a tree $\Delta^{(1)}$ consisting of a single node connected to m leaves, corresponding to letters of input alphabet A. Then, pick a leaf corresponding to a letter with highest probability and replace it with a node connected to m new leaves. Repeat this process successively, picking at each step a leaf corresponding to a word with highest probability. It can be seen that after i steps this algorithm produces a tree $\Delta^{(i)}$ with $(m-1)i+1$ leaves (corresponding to a prefix-free set of words $X(\Delta^{(i)})$) which can be easily enumerated and mapped into $\lceil \lg((m-1)i+1) \rceil$-digit codes.

In Figure 1 we show an example of a code produced by Tunstall algorithm for a binary memoryless source with $P(1) = 0.2$ after 16 iterations.

Tunstall algorithm has been very well studied and found a number of applications in coding theory and beyond. Simple bounds for its redundancy have been independently obtained by Khodak [7] and Jelinek and Schneider [6]. Generalizations of Tunstall code for sources with memory have been proposed by Tjalkens and Willems [18] and Savari and Gallager [13]. More accurate asymptotic analysis of its redundancy (4) has been offered by Savari [14], and most recently, by Drmota, Reznik, Savari, and Szpankowski [5]. Applications of Tunstall algorithm for approximation of uniform distributions, random number generation, and related problems, have been discussed in [2].

Nevertheless, we must stress that Tunstall algorithm is not the only technique available for construction of variable-to-fixed-length codes. Thus, in 1969 G. L. Khodak has suggested the following construction procedure [7] (see also [8], [10]). Start with a tree containing a single node and grow it progressively until all its leaves x satisfy:

$$\frac{1}{N} \leqslant P(x) < \frac{1}{N\,p_{\min}}\,, \tag{5}$$

Code	Word	Pr
0	11	0.04
1	011	0.032
2	101	0.032
3	0001	0.1024
4	0010	0.1024
5	0100	0.1024
6	1000	0.1024
7	0011	0.0256
8	0101	0.0256
9	1001	0.0256
10	00001	0.08192
11	000001	0.065536
12	0000001	0.0524288
13	00000001	0.041943
14	000000001	0.0335544
15	0000000000	0.107374
16	0000000001	0.0268435

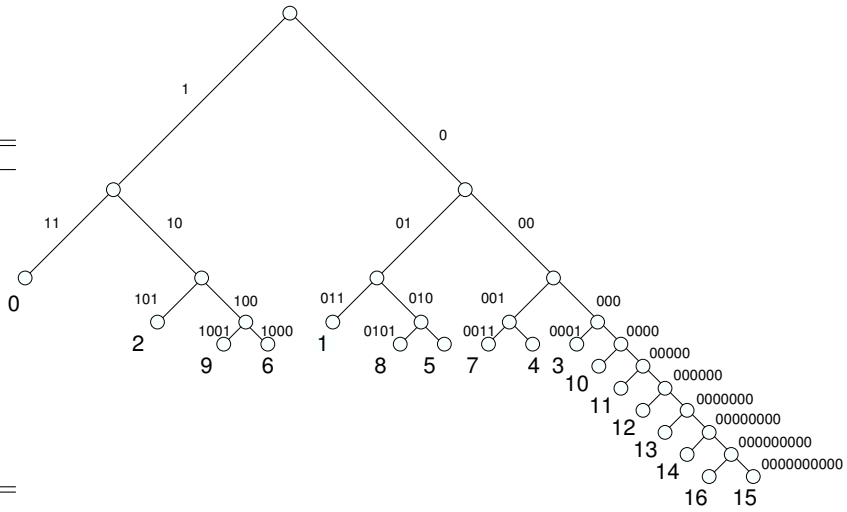

Figure 1: Example VF-code for binary source with $P(1) = 0.2$ (obtained by using either Tunstall algorithm with 16 iterations, or Khodak algorithm with N=40).

where N is a fixed real number ($N > 1/p_{\min}$).

Condition (5) implies that the total number of leaves in the resulting tree Δ_{N} satisfies:

$$|X(\Delta_{\mathrm{N}})| \leqslant N\,, \tag{6}$$

so by picking different values of N one can control the size of the resulting trees (although with certain granularity, since (5) applies to all leaves). It can also be shown (e.g. using [6, Lemma 6]), that a tree Δ_{N} constructed by Khodak algorithm with parameter N is exactly the same as the tree $\Delta^{(i_N)}$ constructed by Tunstall algorithm after $i_N = \frac{|X(\Delta_{\mathrm{N}})|-1}{m-1}$ steps.

In other words, both algorithms can be used to solve the same problem. Tunstall algorithm has an advantage of providing an explicit control over the number of leaves in the resulting trees. Khodak scheme, on the other hand, provides a simple algebraic characterization of the coding tree (5), which, as we will show in this paper, can be used to perform its direct construction, and leads to exact formulae of redundancy of such codes.

3 Enumeration of Nodes in VF Coding Trees

We first prove two simple lemmas allowing us directly enumerate internal nodes in VF-coding trees.

Lemma 1. *Probabilities of internal nodes w in a tree Δ_N satisfying Khodak condition (5) have the following property:*

$$P(w) \geqslant \frac{1}{N\,p_{\min}} \tag{7}$$

Proof. Follows directly from condition (5) and the fact that $P(w)$ must be greater than the probability of a leaf. □

Lemma 2. *Any string $w \in A^*$ such that:*

$$P(w) \geqslant \frac{1}{N\,p_{\min}} \tag{8}$$

leads to an existing internal node in a tree Δ_N satisfying Khodak condition (5).

Proof. If this is not correct, then there must exist a prefix u: $w = uv$, $|u| > 0$, $|v| > 0$, leading to an external node. Then, according to (5) we must have $P(u) < \frac{1}{N\,p_{\min}}$, which, however, contradicts (8) and the fact that $P(w) = P(u)P(v) < P(u)$. □

Next, we derive somewhat tighter bounds for probabilities of leaves in trees constructed using Khodak's algorithm.

Lemma 3. *Leaves x attached to α-branches ($\alpha \in A$) in a tree Δ_N satisfying condition (5) have the following properties:*

$$\frac{P(\alpha)}{N\,p_{\min}} \leqslant P(x) = P(w\alpha) < \frac{1}{N\,p_{\min}} \tag{9}$$

Proof. Consider an internal node corresponding to a string w. Its children correspond to single-letter extensions of w: $w\alpha$, $\alpha \in A$. If now, a node $w\alpha$ becomes an external node, then, according to Khodak condition (5) it must satisfy:

$$P(w\alpha) = P(w)P(\alpha) < \frac{1}{N\,p_{\min}}. \tag{10}$$

Since w is internal (cf. Lemma 1):

$$P(w) \geqslant \frac{1}{N\,p_{\min}}\,. \tag{11}$$

Combining both we arrive at (9). □

It turns out that this condition is sufficient for direct counting of leaves.

Lemma 4. *All strings $w \in A^*$ such that:*

$$\frac{P(\alpha)}{N\,p_{\min}} \leqslant P(w\alpha) < \frac{1}{N\,p_{\min}} \tag{12}$$

correspond to internal nodes whose α-branches are immediately connected to leaves in a tree Δ_N satisfying Khodak condition (5).

Proof. Left side of (12) ensures that w is a valid internal node in Δ_N (cf. Lemma 2). It also implies that (5) is satisfied for $w\alpha$, so it must be a leaf. □

Remark 1. Note, that while original Khodak's condition (5) is formulated as limits on probabilities of leaves, it cannot be used for their direct enumeration. In order to use (5), one has to maintain a tree structure and grow it progressively until all its leaves satisfy (5). Our condition (12) can be used directly, without building a tree structure.

4 Structure of VF Coding Trees

Lemma 4 implies that all words in a VF-code Δ_N for memoryless source can be enumerated as follows:

$$X\left(\Delta_N\right) = \bigcup_{\substack{k_1+\ldots+k_m=\ell,\,i \\ \frac{p_i}{N p_{\min}} \leqslant p_1^{k_1}\ldots p_m^{k_m} p_i < \frac{1}{N p_{\min}}}} X_{(\ell,k_1,\ldots,k_m),i} \tag{13}$$

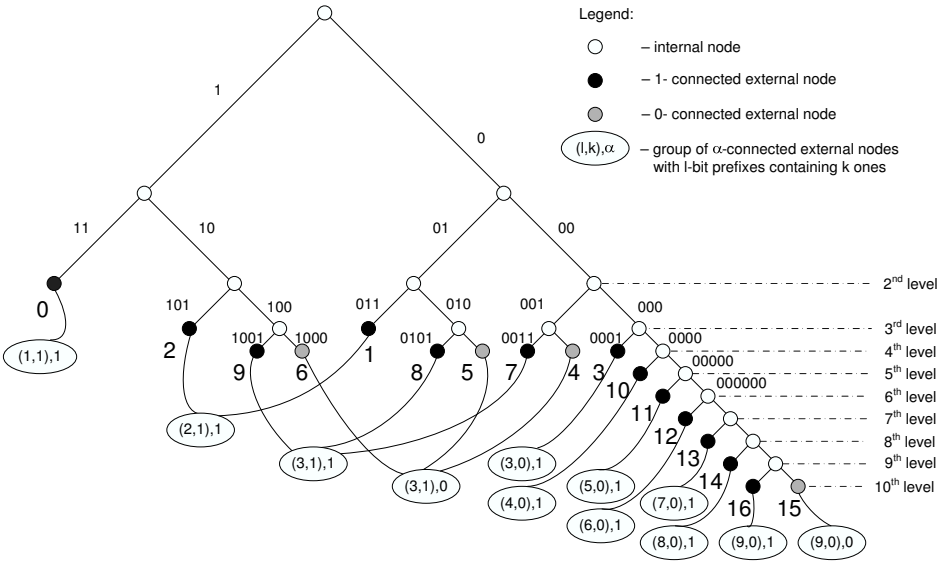

Figure 2: Structure of an example VF-code for memoryless source.

where each group $X_{(\ell,k_1,\ldots,k_m),i}$, contains $\binom{\ell}{k_1,\ldots,k_m}$ words of length $\ell+1$, terminating on i-th symbol, and having probabilities $p_1^{k_1}\ldots p_m^{k_m}p_i$, where $k_1,\ldots,k_m$ indicate numbers of symbols of each kind in their first ℓ positions.

In a binary case, with $P(1)=p, P(0)=q, p<q$, equation (13) can be rewritten as:

$$X\left(\Delta_N\right) = \bigcup_{\substack{0\leqslant k\leqslant \ell,\,\alpha \\ \frac{P(\alpha)}{Np}\leqslant p^k q^{\ell-k}P(\alpha)<\frac{1}{Np}}} X_{(\ell,k),\alpha} \tag{14}$$

and where groups $X_{(\ell,k),\alpha}$ are formed by $\binom{\ell}{k}$ words of length $\ell+1$, in which first ℓ-bits contain k ones, and the last symbol is α.

We illustrate the structure of such a partition in Fig. 2. This VF-code is identical to our previous example (see Fig. 1).

It can be observed that for each level ℓ, the number of groups $X_{(\ell,k),\alpha}$ belonging to this level is at most $2\left(\ell+1\right)$ (which corresponds to a case when the tree is a complete tree with all leaves located at level ℓ). Hence, the total number of groups containing in L levels of the tree Δ_N is at most $O\left(L^2\right)$. This is significantly less than the total number of nodes or leaves containing in a tree of the same depth (which is typically exponential: $O\left(2^L\right)$).

5 Enumerative Construction of VF-Codes

Using the above observations we can now design an algorithm for enumerative construction of VF-codes. An example of a possible compact data structure and C-code of a construction procedure for binary memoryless sources is given in Algorithm 1.

The data structure used by Algorithm 1, stores only parameters ℓ, k, α of each group $X_{(\ell,k),\alpha}$ and parameter *offset*, representing the value of the first codeword in a group. As discussed earlier, the size of such a representation of VF-coding tree in computer's memory is at most $O(L^2)$, where L is the height of a tree.

Algorithm 1 VF-code group structure and algorithm for direct construction of VF-codes.

```
/* VF-Code Group structure: */
typedef struct {
  unsigned char l,k,a;  /* l - prefix length; k - # of ones; a - last symbol */
  unsigned int  offset; /* code of the lexicographically smallest word in the group */
} VFCG;

int make_code (double p, double N, int L, VFCG *vfcg)
{
  int k,l,j=0,M=0;              // j = group #; M = # of codewords
  for (l=0; l<=L; l++) {        // l = length of words w; L = max. length
    for (k=0; k<=l; k++) {      // k = number of ones in w
      /* compute probability of an l-bit word with k ones: */
      double pr = pow(p,k) * pow(1-p,l-k);
      /* internal nodes? */
      if (pr >= 1/(N * p)) {
        /* check if left extensions turn into leaves: */
        if (pr < 1/(N * p * p)) {
          /* register an l,k,0 group: */
          vfcg[j].l = l; vfcg[j].k = k; vfcg[j].a = 0;
          vfcg[j].offset = M;
          j ++;
          M += binomial(l,k);
        }
        /* check if right extensions turn into leaves: */
        if (pr < 1/(N * p * (1-p))) {
          /* register an l,k,1 group: */
          vfcg[j].l = l; vfcg[j].k = k; vfcg[j].a = 1;
          vfcg[j].offset = M;
          j ++;
          M += binomial(l,k);
        }
      }
    }
  }
  /* set end-of-list record & return # of codewords */
  vfcg[j].offset = M;
  return M;
}
```

Algorithms 2 and 3 present encoding and decoding procedures using this data structure. For the purpose of illustration of the idea, in both cases we employ simple linear search for matching group. The number of search items in both cases is at most $O(L^2)$.

Practical implementations may employ more sophisticated searching algorithms, which could result in much faster decoding. For example, the use of binary search should result in only $O(\log L)$ steps, which is much faster that $O(L)$ steps needed for parsing of the original VF-coding tree.

In order to compute a code for a word w belonging to a group $X_{(\ell,k),\alpha}$, Algorithm 2 computes *lexicographic index* of ℓ-bits prefix of w in a set of all ℓ-bits long words with k ones. This procedure in Algorithm 2 is denoted as $index(\ell, k, w)$. Similarly, Algorithm 3 uses the reverse process, denoted as $word(\ell, k, i)$, which generates i-th word from $X_{(\ell,k),\alpha}$.

Algorithm 2 Encoding of a VF-code using list of groups in a coding tree.

```
/* Variable-to-fixed-length encoder: */
void vfc_encode (BITSTREAM *in, unsigned *code, VFCG *vfcg)
{
  unsigned l, k, j, i, w;
  for (l=0,k=0,j=0,w=0; ; ) {
    i = get_bit(in);                          // read next bit
    while (l == vfcg[j].l) {                  // see if we have a match
      if (k == vfcg[j].k && i == vfcg[j].a)
        goto found;
      j ++;
    }
    w = w *2 + i; k += i; l ++;               // include l-th bit in prefix
  }
found:
  i = index(l,k,w);                           // get lexicographic index of the prefix
  *code = vfcg[j].offset + i                  // compute codeword
}
```

Algorithm 3 Decoding of a VF-code using list of groups in a coding tree.

```
/* Variable-to-fixed-length decoder: */
void vfc_decode (unsigned code, BITSTREAM *out, VFCG *vfcg)
{
  unsigned i, j, w;
  for (j=0; code > vfcg[j+1].offset; j++) ; // find a subgroup containing the code
  i = code - vfcg[j].offset;                // i = lexicographic index of a prefix
  w = word(vfcg[j].l,vfcg[j].k,i);          // generate i-th word in an (l,k) set
  w = w * 2 + vfcg[j].a;                    // append last bit
  put_bits(w, vfcg[j].l+1, out);            // output decoded word
}
```

We note that for reasonably short words (e.g. $\ell \leqslant 10 \ldots 16$) computation of their lexicographic indices (or synthesis of words, using their indices), can be a matter of a single lookup. For longer words, we can use the following well-known combinatorial formula (cf. [11], [1], [15], [3], [19], [20]):

$$index(\ell, k, w) = \sum_{j=1}^{\ell} w_j \binom{\ell - j}{\sum_{k=j}^{\ell} w_k}, \tag{15}$$

where w_j represent individual bits of the word w, and it is assumed that $\binom{\ell}{k} = 0$ for all $k > \ell$. In order to implement it, one could either pre-compute all binomial coefficients up to level ℓ in Pascal's triangle, or compute them dynamically, using the following simple identities:

$$\binom{\ell - k}{k - 1} = \frac{k}{\ell}\binom{\ell}{k}, \quad \text{and} \quad \binom{\ell - k}{k} = \frac{\ell - k}{\ell}\binom{\ell}{k}.$$

The implementation based on pre-computed coefficients requires $\frac{\ell\,(\ell+1)}{2} = O\left(\ell^2\right)$ words of memory, and $O(\ell)$ additions. Dynamic computation of coefficients will require $O(\ell)$

additions, multiplications and divisions, but the entire process needs only few registers. Additional discussion on complexity of index computation can be found in [20].

6 Exact Redundancy of VF-codes

In this section we claim the following result:

Theorem 1. *VF-codes based on a tree Δ_N satisfying Khodak condition (5) have the following redundancies:*

$$R_{\mathrm{VF}}(\Delta_N, S) = \frac{\lceil \lg (M(N,S)) \rceil}{d(N,S)} - h(S)\,, \quad R^*_{\mathrm{VF}}(\Delta_N, S) = \frac{\lg (M(N,S))}{d(N,S)} - h(S)\,, \tag{16}$$

where:

$$d(N,S) = \sum_{\ell} \sum_{\substack{k_1+\ldots+k_m=\ell \\ p_1^{k_1}\ldots p_m^{k_m} \geqslant \frac{1}{N\,p_{\min}}}} \binom{\ell}{k_1,\ldots,k_m} p_1^{k_1}\ldots p_m^{k_m}\,, \tag{17}$$

and

$$M(N,S) = 1 + (m-1)\sum_{\ell} \sum_{\substack{k_1+\ldots+k_m=\ell \\ p_1^{k_1}\ldots p_m^{k_m} \geqslant \frac{1}{N\,p_{\min}}}} \binom{\ell}{k_1,\ldots,k_m}. \tag{18}$$

Proof. By $X(\Delta)$ and $W(\Delta)$ we denote sets of strings leading to all external and internal nodes in Δ, correspondingly. To derive an expression for the average delay, we use the fact that [16, 10]:

$$d(\Delta, S) = \sum_{x \in X(\Delta)} P(x)|x| = \sum_{w \in W(\Delta)} P(w)\,, \tag{19}$$

and then simply enumerate all internal nodes using condition (8) of Lemma 2:

$$\begin{aligned} d(N,S) \;:=\; d(\Delta_N, S) \;&=\; \sum_{w \in A^*:\, P(w) \geqslant \frac{1}{N\,p_{\min}}} P(w) \\ &=\; \sum_{\ell} \sum_{\substack{k_1+\ldots+k_m=\ell \\ p_1^{k_1}\ldots p_m^{k_m} \geqslant \frac{1}{N\,p_{\min}}}} \binom{\ell}{k_1,\ldots,k_m} p_1^{k_1}\ldots p_m^{k_m}\,. \end{aligned} \tag{20}$$

Similarly, to count the number of external nodes, we use the fact that:

$$|X(\Delta)| = 1 + (m-1)\,|W(\Delta)|\,, \tag{21}$$

and apply Lemma 2:

$$\begin{aligned} M(N,S) \;:=\; |X(\Delta_N)| \;&= 1 + (m-1) \sum_{w \in A^*:\, P(w) \geqslant \frac{1}{N\,p_{\min}}} 1 \\ &=\; 1 + (m-1)\sum_{\ell} \sum_{\substack{k_1+\ldots+k_m=\ell \\ p_1^{k_1}\ldots p_m^{k_m} \geqslant \frac{1}{N\,p_{\min}}}} \binom{\ell}{k_1,\ldots,k_m}. \end{aligned} \tag{22}$$

□

6.1 Exact Redundancy in Binary Case

In binary case, assuming that $P(1) = p, P(0) = q, p < q$, the condition (8) becomes

$$P(w) = p^k q^{\ell-k} \geqslant \frac{1}{Np} \tag{23}$$

where k denotes the number of ones in the path from root to node w in the parsing tree, and ℓ is the total length of this path. The above inequality implies that:

$$k \leqslant -\frac{\log(Np) + \ell \log q}{\log(p/q)} \tag{24}$$

and (by requiring $k \geqslant 0$):

$$\ell \leqslant -\frac{\log(Np)}{\log q}\,. \tag{25}$$

Using these expressions we obtain:

Corollary 1. *VF-codes for binary memoryless source with probabilities p, q $(p < q)$ have the following redundancies:*

$$R_{\mathrm{VF}}(\Delta_N, p) = \frac{\lceil \lg(M(N,p)) \rceil}{d(N,p)} - h\,, \quad R^*_{\mathrm{VF}}(\Delta_N, p) = \frac{\lg(M(N,p))}{d(N,p)} - h\,, \tag{26}$$

where:

$$d(N,p) = \sum_{0 \leqslant \ell \leqslant \left\lfloor -\frac{\log(Np)}{\log q} \right\rfloor} \; \sum_{0 \leqslant k \leqslant \min\left\{\ell, \left\lfloor \frac{\log(Np) + \ell \log q}{\log(q/p)} \right\rfloor\right\}} \binom{\ell}{k} p^k q^{\ell-k}\,, \tag{27}$$

$$M(N,p) = 1 + \sum_{0 \leqslant \ell \leqslant \left\lfloor -\frac{\log(Np)}{\log q} \right\rfloor} \; \sum_{0 \leqslant k \leqslant \min\left\{\ell, \left\lfloor \frac{\log(Np) + \ell \log q}{\log(q/p)} \right\rfloor\right\}} \binom{\ell}{k}\,, \tag{28}$$

and $h = -p \lg p - q \lg q$ is the entropy of the source.

The obtained expressions (27) and (28) are well suited for evaluations using computers. Examples of such computations (conducted using MAPLE) for binary sources with $p = \frac{2}{5}$ (a case when $\frac{\ln p}{\ln q}$ is irrational) and $p = \frac{2}{3+\sqrt{5}}$ (a case when $\frac{\ln p}{\ln q} = 2$) are shown in Fig. 2. For comparison, in Fig. 2 we also include plots of idealized redundancy computed using known asymptotic formulae [5].

Based on these plots it can be seen that exact redundancy of VF-codes $R_{\mathrm{VF}}(M,p)$ involves significant fluctuations caused by rounding of codeword lengths to integral number of bits. While the magnitude of those oscillations is decreasing as $O(1/\log M)$, they are of major effect when the size of VF-coding trees is constrained. In such cases our exact redundancy formula can be used as a tool for optimal design of such codes.

In contrast, the idealized redundancy $R^*_{\mathrm{VF}}(M,p)$ has a much smaller amplitude of oscillations in a case when $\frac{\ln p}{\ln q}$ is irrational, and it is oscillation-free in the rational case. In both cases, asymptotic formulae from [5] provide very good approximations to the exact values of $R^*_{\mathrm{VF}}(M,p)$.

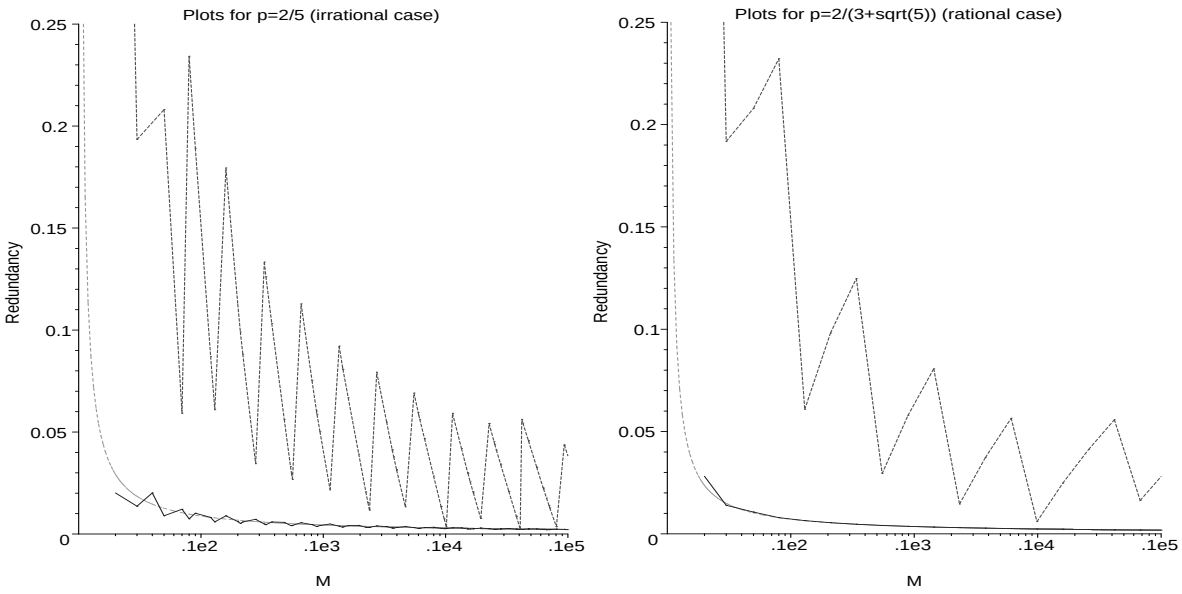

Figure 3: Exact average redundancy $R_{\mathrm{VF}}(M,p)$ (dashed lines) vs. exact idealized redundancy $R^*_{\mathrm{VF}}(M,p)$ (solid lines) vs asymptotic idealized redundancy (dotted lines) of VF-codes constructed for binary memoryless sources with $p=\frac{2}{5}$ and $p=\frac{2}{3+\sqrt{5}}$.

References

[1] V. F. Babkin, A method of universal coding with non-exponent labour consumption, *Probl. Inf. Trans.*, 1 (4) (1971) 13–21 (in Russian)

[2] F. Cicalese, L. Gargano, and U. Vaccaro, A note on Tunstall codes with applications to optimal approximation of uniform distributions, *IEEE Trans. Inf. Theory* – submitted.

[3] T. M. Cover, Enumerative Sources Encoding, *IEEE Trans. Inf. Theory*, 19 (1) (1973) 73–77.

[4] T. M. Cover and J. M. Thomas, *Elements of Information Theory*, (John Wiley & Sons, New York, 1991).

[5] M. Drmota, Y. A. Reznik, S. A. Savari, and W. Szpankowski, Precise Asymptotic Analysis of the Tunstall Code, IEEE International Symposium on Information Theory (ISIT06), Seattle, WA, July 9 - 14, 2006 – accepted.

[6] F. Jelinek, and K. S. Schneider, On Variable-Lenght-to-Block Coding, *IEEE Trans. Inf. Theory*, **18** (6) (1972) 765–774.

[7] G. L. Khodak, Connection Between Redundancy and Average Delay of Fixed-Length Coding, *All-Union Conference on Problems of Theoretical Cybernetics* (Novosibirsk, USSR, 1969) 12 (in Russian)

[8] G. L. Khodak, Redundancy Estimates for Word-Based Encoding of Messages Produced by Bernoulli Sources, *Probl. Inf. Trans.*, **8**, (2) (1972) 21–32 (in Russian).

[9] D. Knuth, *The Art of Computer Programming. Sorting and Searching. Vol. 3* (Addison-Wesley, Reading MA, 1973).

[10] R. E. Krichevsky, Universal Data Compression and Retrieval. (Kluwer, 1993).

[11] V. I. Mudrov, An algorithm for enumeration of combinations, *Vyc. Math. and Math. Phys.*, 5 (4) (1965) 776–778 (in Russian).

[12] B. Ya. Ryabko, The fast enumeration of combinatorial objects, *Discrete Math.and Applications*, **10**, (2), 1998.

[13] S. A. Savari, Robert G. Gallager; Generalized Tunstall codes for sources with memory, IEEE Trans. Info. Theory, vol. IT-43, pp. 658 - 668, March 1997.

[14] S. A. Savari, Variable-to-Fixed Length Codes for Predictable Sources, Proc IEEE Data Compression Conference, Snowbird, UT, March 30 - April 1, 1998, pp. 481–490.

[15] J. P. M. Schalkwijk, An Algorithm For Source Coding, *IEEE Trans. Inf. Theory*, 18 (3) (1972) 395–399.

[16] V. M. Sidelnikov, On Statistical Properties of Transformations Carried out by Finite Automata, *Cybernetics*, 6 (1965) 1–14 (in Russian)

[17] Tj. J. Tjalkens, Efficient and fast data compression codes for discrete sources with memory, Ph.D. Thesis, Eindhoven Univ. Techn., The Netherlands, 1987.

[18] T. J. Tjalkens and F. M. J. Willems, "Variable to fixed-length codes for Markov sources," *I.E.E.E. Trans. Inf. Theory* IT-33, 246-257, 1987.

[19] Tj.J. Tjalkens, The Complexity of Minimum Redundancy Coding, *in Proc. 21-th Symp. Inf. Theory in the Benelux* (May 2000) 247-254.

[20] T. Tjalkens, Implementation cost of the Huffman-Shannon-Fano code, *in Proc. Data Compression Conference (DCC'05)* (Snowbird, Utah, March 29-31, 2005) 123–132.

[21] B. P. Tunstall, Synthesis of Noiseless Compression Codes, Ph.D. dissertation,(Georgia Inst. Tech., Atlanta, GA, 1968)