Rényi to Rényi — Source Coding under Siege

Michael B. Baer
Electronics for Imaging
303 Velocity Way
Foster City, California 94404 USA
Email: Michael.Baer@efi.com

***Abstract*— A novel lossless source coding paradigm applies to problems of unreliable lossless channels with low bitrate, in which a vital message needs to be transmitted prior to termination of communications. This paradigm can be applied to Alfréd Rényi's secondhand account of an ancient siege in which a spy was sent to scout the enemy but was captured. After escaping, the spy returned to his base in no condition to speak and unable to write. His commander asked him questions that he could answer by nodding or shaking his head, and the fortress was defended with this information. Rényi told this story with reference to traditional lossless source coding, in which the objective is minimization of expected codeword length. The goal of maximizing probability of survival in the siege scenario is distinct from yet related to this traditional objective. Rather than finding a code minimizing expected codeword length $\sum_{i=1}^{n} p(i)l(i)$, this variant involves maximizing $\sum_{i=1}^{n} p(i)\theta^{l(i)}$ for a known $\theta \in (0,1)$. When there are no restrictions on codewords, this problem can be solved using a known generalization of Huffman coding. The optimal solution has coding bounds which are functions of Rényi entropy; in addition to known bounds, new bounds are derived here. The alphabetically constrained version of this problem has applications in search trees and diagnostic testing. A novel dynamic programming algorithm — based upon the oldest known algorithm for the traditional alphabetic problem — optimizes this problem in $O(n^3)$ time and $O(n^2)$ space, whereas two novel approximation algorithms can find a suboptimal solution faster: one in linear time, the other in $O(n \log n)$. Coding bounds for the alphabetic version of this problem are also presented.**

I. Introduction

Alfred Rényi related an ancient scenario in which the Romans held rebels under siege, rebels whose only hope was the knowledge gathered by a mute, illiterate spy, one who could only nod and shake his head [1, pp. 13-14]. This apocryphal tale — based upon a historical siege — is the premise behind the Hungarian version of the spoken parlor game Twenty Questions. A modern parallel in the 21st century occurred when Russian forces gained the knowledge needed to defeat hostage-takers by asking hostages "yes" or "no" questions over mobile phones [2], [3].

Rényi presented this problem in narrative form in order to motivate the relation between Shannon entropy and binary source coding. Note however that Twenty Questions, source coding, and the siege scenario actually have three different objectives. In Twenty Questions, the goal is to be able to determine an item (or message) by asking at most twenty questions. In source coding, the goal is to minimize the expected number of questions — or, equivalently, bits — necessary to determine the message. For the siege scenario, the goal is survival, that is, assuming partial information is not useful, the besieged would wish to maximize the probability that the message is successfully transmitted within a certain window of opportunity. When this window closes and the siege ends, the information becomes worthless. An analogous situation occurs when a wireless device is temporarily within range of a base station; one can safely assume that the channel, when available, will transmit at the lowest (constant) bitrate, and will be lost at a nondeterministic time after its availability.

We consider this modified source coding problem and derive properties of and algorithms for the optimization of the problem and variants thereof. In Section II, we formalize the problem and find its solution in a generalization of the Huffman coding algorithm previously used for a complementary problem. Section III concerns several extensions and variants of the problem. In particular, restricting the solution space to alphabetic codes is considered in Section IV, with a dynamic programming algorithm presented for optimizing the alphabetic code, one that extends to the related problem of search trees. In Section V, we consider entropy bounds in the form of Rényi entropy for the unrestricted problem, leading to a new bound and a related property involving the length of the shortest codeword of an optimal code. Entropy bounds for the alphabetic problem, along with linear-time approximation algorithms, are derived in Section VI. Section VII concludes with related work and a possible future direction.

II. Formalizing the Problem

A message is represented by symbol X drawn from the alphabet $\mathcal{X} \triangleq \{1, 2, \ldots, n\}$. Symbol i has probability $p(i)$, defining probability mass function p, known to both sender and receiver. The source symbols are coded into binary codewords, each bit of which is equivalent to an answer to a previously agreed-upon "yes" or "no" question; the meaning of each question (bit context) is implied by the previous answers (bits), if any, in the current codeword. Each codeword $c(i)$, corresponding to symbol i, has length $l(i)$, defining overall length vector $\boldsymbol{l}$ and overall code C.

Let $\mathcal{L}_n$ be the set of allowable codeword length vectors, those that satisfy the Kraft inequality, that is,

$$\mathcal{L}_n \triangleq \left\{ \boldsymbol{l} \in \mathbb{Z}_+^n \text{ such that } \sum_{i=1}^{n} 2^{-l(i)} \leq 1 \right\}.$$

Furthermore, assume that the duration of the window of opportunity is independent of the communicated message

and is memoryless. Memorylessness implies that the window duration is distributed exponentially. Therefore, quantizing time in terms of the number of bits T that we can send within our window,

$$P(T = t) = (1 - \theta)\theta^t,\ t = 0, 1, 2, \ldots$$

with known parameter $\theta < 1$. We then wish to maximize the probability of success, i.e., the probability that the message length does not exceed the quantized window length:

$$\begin{aligned} P[l(X) \le T] &= \sum_{t=0}^{\infty} P(T = t) \cdot P[l(X) \le t] \\ &= \sum_{t=0}^{\infty} (1 - \theta)\theta^t \cdot \sum_{i=1}^{n} p(i) 1_{l(i) \le t} \\ &= \sum_{i=1}^{n} p(i) \cdot (1 - \theta) \sum_{t=l(i)}^{\infty} \theta^t \\ &= \sum_{i=1}^{n} p(i)\theta^{l(i)} \cdot (1 - \theta) \sum_{t=0}^{\infty} \theta^t \\ &= \sum_{i=1}^{n} p(i)\theta^{l(i)} \end{aligned}$$

The problem is thus the following optimization:

$$\max_{\boldsymbol{l} \in \mathcal{L}_n} P[l(X) \le T] = \max_{\boldsymbol{l} \in \mathcal{L}_n} \sum_{i=1}^{n} p(i)\theta^{l(i)} \tag{1}$$

To maximize this probability of success, we use a generalization of Huffman coding developed independently by Hu *et al.* [4, p. 254], Parker [5, p. 485], and Humblet [6, p. 25], [7, p. 231]. The bottom-up algorithm of Huffman coding starts out with n weights of the form $w(i) = p(i)$ and combines the two least probable symbols x and y into a two-node subtree; for algorithmic reduction, this subtree of combined weights is subsequently considered as one symbol with weight (combined probability) $w(x)+w(y)$. (We use the term "weights" because one can turn a problem of rational probabilities into one of integer weights for implementation.) Reducing the problem to one with one fewer item, the process continues recursively until all items are combined into a single code tree. The generalization of Huffman coding used to maximize (1) instead assigns the weight

$$\theta \cdot (w(x) + w(y))$$

to the root node of the subtree of merged items. With this modified combining rule, the algorithm proceeds in a similar manner as Huffman coding, yielding a code with optimal probability of success.

III. Related Problems

Note that if we use this probability of success as a tie-breaker among codes with minimal expected length — those optimal under the traditional measure of coding — the solution is unique and independent of the value of θ, a straightforward consequence of [8]. We can obtain this optimal code by using the top-merge variation of Huffman coding given in [9]; this variation views combined items as "smaller" than individual items of the same weight. Similarly, for θ sufficiently near 1 — i.e., if the amount of information to be communicated is large compared to the size of the window in question — the optimal solution is identical to this top-merge solution, a straightforward result analogous to that noted in [10, p. 222]. Thus traditional Huffman coding should be used if the window size is expected to be far larger than the message size.

Observe also that, if we change the probability of $P(T = 0)$ without changing the ratios between the other probabilities, the problem's solution code does not change, even though the probability of success does. There are still more criteria that are identically optimized, including, if we have several independent messages serially transmitted, maximizing the number of messages expected to be sent within a window. Another problem arises if we have a series of windows with independent instances of the problem and want to minimize the expected numbers of windows needed for success. The maximization of probability minimizes this number, which is the inverse of the probability of success in each window:

$$E[N_{\text{indep}}] = \left(\sum_{i=1}^{n} p(i)\theta^{l(i)} \right)^{-1}$$

Note that this is a risk-loving objective, in that we are more willing than in standard coding to trade off having longer codewords for unlikely items for having shorter codewords for likely items.

However, if the message to send is constant across all windows rather than independent, the expected number of windows needed — assuming it is necessary to restart communication for each window — is instead

$$E[N_{\text{const}}] = \sum_{i=1}^{n} p(i)\theta^{-l(i)}.$$

This is a risk-averse objective, in that we are less willing to make the aforementioned tradeoff than in standard coding. These distinct objective functions can be combined into one if we normalize, that is, if we seek to minimize penalty function

$$L_\theta(p, \boldsymbol{l}) \triangleq \log_\theta \sum_{i=1}^{n} p(i)\theta^{l(i)} \tag{2}$$

for $\theta > 0$, where minimizing expected length is the limit case of $\theta \to 1$. Campbell first noticed this in [11]. Others later found that the aforementioned generalized Huffman-like algorithm optimizes this for all $\theta > 0$, though previously only $\theta \ge 1$ had any known application.

IV. Alphabetic Codes

Under siege, assuming the absence of a predetermined code, using the optimal Huffman-like code would likely be impractical, since one would need to account not only for the time taken to answer a question, but the time needed to ask it. In this, and in applications such as search trees and testing for faulty devices in a sequential input-output system [12] —

assuming the answer remains binary — each question should be of the form, "Is the output greater than x?" where x is one of the possible symbols, a symbol we call the *splitting point* for the corresponding node. This restriction is equivalent to the constraint that $c(j) \prec c(k)$ whenever $j < k$, where codewords $c(\cdot)$ are compared using lexicographical order. The dynamic programming algorithm of Gilbert and Moore [13] can be adapted to this restricted problem.

The key to the modified algorithm is to note that any optimal coding tree must have all its subtrees optimal. Since there are $n-1$ possible splitting points, if we know all potential optimal subtrees for all possible ranges, the splitting point can be found through sequential search of the possible combinations. The optimal tree is thus found inductively, and this algorithm has $O(n^3)$ time complexity and $O(n^2)$ space complexity. The dynamic programming algorithm involves finding the maximum tree weight $W_{j,k}$ (and corresponding optimum tree) for items j through k for each value of $k-j$ from 0 to $n-1$, computing inductively, starting with $W_{j,j} = w(j)$ $(= p(j))$, with

$$W_{j,k} = \theta \max_{s \in \{j, j+1, \ldots, k-1\}} [W_{j,s} + W_{s+1,k}]$$

for $j < k$. Knuth showed how the traditional linear version of this approach can be extended to general search trees [14]; for the siege scenario, this is a straightforward generalization, which we omit here for brevity. For answers having unequal cost, algorithms analogous to the linear-objective ones given in [15], [16] are similarly formulated.

Another contribution of Knuth in [14] was to reduce algorithmic complexity for the linear version using the fact that the splitting point of an optimal tree must be between the splitting points of the two (possible) optimal subtrees of size $n'-1$. With the siege problem, this property no longer holds; a counterexample to this is $\theta = 0.6$ with weights $(8, 1, 9, 6)$.

Similarly, for the linear problem [17], as well as for $\theta > 1$ and some nonexponential problems [4], there is a well-known procedure — the Hu-Tucker algorithm — for finding an optimal alphabetic solution in $O(n \log n)$ time and linear space. The corresponding algorithm for $\theta < 1$ fails, however, this time for $\theta = 0.6$ and weights $(8, 1, 9, 6, 2)$. Approximation algorithms presented in Section VI, though, have similar or lesser complexity.

V. Bounds on Optimal Codes

Returning to the general (nonalphabetic) case, it is often useful to come up with bounds on the performance of the optimal code. In this section, we assume without loss of generality that $p(1) \geq p(2) \geq \cdots \geq p(n)$. Note that $\theta \leq 0.5$ is a trivial case, always solved by a unary code, $C_{\mathrm{u}} \triangleq (0, 10, 110, \ldots, 11\cdots10, 11\cdots11)$. For nontrivial $\theta > 0.5$, there is a relationship between the problem and Rényi entropy.

Campbell first proposed a decaying exponential utility function for coding in [18]. He observed a simple upper bound for (1) with $\theta > 0.5$ in [18] and alluded to a lower bound in [19]. These bounds are similar to the well-known Shannon entropy bounds for Huffman coding (e.g., [20, pp. 87-88], [21]). In this case, however, the bounds involve Rényi's α-entropy [22], not Shannon's. Rényi entropy is

$$H_\alpha(p) \triangleq \frac{1}{1-\alpha} \log_2 \sum_{i=1}^{n} p(i)^\alpha$$

where, in this case,

$$\alpha \triangleq \frac{1}{\log_2 2\theta} = \frac{1}{1 + \log_2 \theta}.$$

For nontrivial maximizations ($\theta \in (0.5, 1)$),

$$\theta^{H_\alpha(p)+1} < \max_{\boldsymbol{l} \in \mathcal{L}_n} P[l(X) \leq T] \leq \theta^{H_\alpha(p)}. \tag{3}$$

We can rephrase this using the definition of $L_\theta(p, \boldsymbol{l})$ in (2) as

$$0 \leq \min_{\boldsymbol{l} \in \mathcal{L}_n} L_\theta(p, \boldsymbol{l}) - H_\alpha(p) < 1, \tag{4}$$

a similar result to the traditional coding bound [21]. Inequality (4) also holds for the minimization problem of $\theta > 1$.

As an example of these bounds, consider the probability distribution implied by Benford's law [23], [24]:

$$p(i) = \log_{10}(i+1) - \log_{10}(i), \;\; i = 1, 2, \ldots 9 \tag{5}$$

At $\theta = 0.9$, for example, $H_\alpha(p) \approx 2.822$, so the optimal code will have between a 0.668 and 0.743 chance of success. Running the algorithm, the optimal lengths are $\boldsymbol{l} = (2, 2, 3, 3, 4, 4, 4, 5, 5)$, resulting in a probability of success of 0.739.

More sophisticated bounds on the optimal solution for the $\theta > 1$ case were given in [25]; these appear as solutions to related problems rather than in closed form. Closed-form bounds given in [26] are functions of entropy (of degree α) and $p(1)$, as in the linear case [27]–[32]. These bounds are flawed, however, in that they assume $p(1) \geq 0.4$ always implies an optimal code exists with $l(1) = 1$. A simple counterexample to this assumption is $p = (0.55, 0.15, 0.15, 0.15)$ with $\theta = 2$, where $l(i) = 2$ for all i.

However, when $\theta < 1$, because the multiplication step of the generalized Huffman-like coding algorithm provides for a strict reduction in weight, $l(1) = 1$ for any $p(1) \geq 0.4$. Here we present better conditions on $l(1) = 1$ and show that they are tight, then derive better entropy bounds from them.

Theorem 1: If $p(1) \geq 2\theta(2\theta+3)^{-1}$, then there is an optimal code for p with $l(1) = 1$.

This is a generalization of [28] and is only slightly more complex to prove:

Proof: Recall that the generalized Huffman algorithm combines the items with the smallest weights, w' and w'', yielding a new item of weight $w = \theta(w' + w'')$, and this process is repeated on the new set of weights, the tree thus constructed up from the leaves to the root. Consider the step at which item 1 gets combined with other items; we wish to prove that this is the last step. At the beginning of this step the (possibly merged) items left to combine are $\{1\}, S_2^k, S_3^k, \ldots, S_k^k$, where we use S_j^k to denote both a (possibly merged) item of weight $w(S_j^k)$ and the set of (single)

items combined to make item S_j^k. Since $\{1\}$ is combined in this step, all but one S_j^k has at least weight $p(1)$. Recall too that all weights $w(S_j^k)$ must be less than or equal to the sums of probabilities $\sum_{i\in S_j^k} p(i)$. Then

$$\begin{array}{rcll} \frac{2\theta(k-1)}{2\theta+3} & \leq & (k-1)p(1) & \\ & < & p(1)+\sum_{j=2}^{k} w(S_j^k) & \\ & \leq & p(1)+\sum_{j=2}^{k}\sum_{i\in S_j^k} p(i) & \\ & = & \sum_{i=1}^{n} p(i) & = \quad 1 \end{array}$$

which, since $\theta > 0.5$, means that $k < 5$. Thus, because $n < 4$ is a trivial case, we can consider the steps in generalized Huffman coding at and after which four items remain, one of which is item $\{1\}$ and the others of which are S_2^4, S_3^4, and S_4^4. We show that, if $p(1) \geq 2/(2\theta+3)$, these items are combined as shown in Fig. 1.

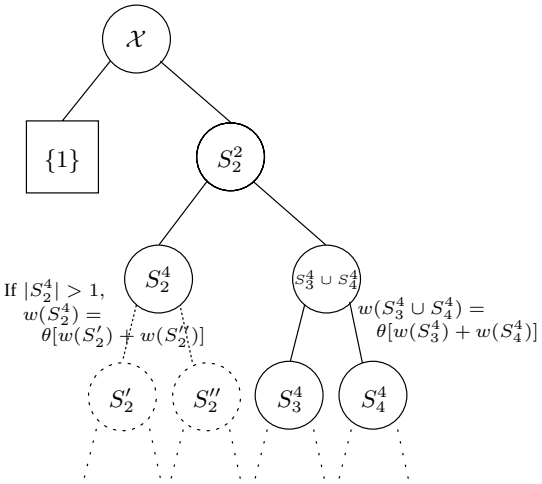

Fig. 1. Tree in last steps of the generalized Huffman algorithm

We assume without loss of generality that weights $w(S_2^4)$, $w(S_3^4)$, and $w(S_4^4)$ are in descending order. From $w(S_2^4) + w(S_3^4) + w(S_4^4) \leq \sum_{i=2}^{n} p(i) \leq 3/(2\theta+3)$, $w(S_2^4) \geq w(S_3^4)$, and $w(S_2^4) \geq w(S_4^4)$, it follows that $w(S_3^4)+w(S_4^4) \leq 2/(2\theta+3)$. Consider set S_2^4. If its cardinality is 1, then $w(S_2^4) \leq p(1)$, so the next step merges the least two weighted items S_3^4 and S_4^4. Since the merged item has weight at most $2\theta/(2\theta+3)$, this item can then be combined with S_2^4, then $\{1\}$, so that $l(1) = 1$. If S_2^4 is a merged item, let us call the two items (sets) that merged to form it S_2' and S_2'', indicated by the dashed nodes in Fig. 1. Because these were combined prior to this step, $w(S_2') + w(S_2'') \leq w(S_3^4) + w(S_4^4)$, so $w(S_2^4) \leq \theta[w(S_3^4) + w(S_4^4)] \leq 2\theta/(2\theta+3)$. Thus $w(S_2^4)$, and by extension $w(S_3^4)$ and $w(S_4^4)$, are at most $p(1)$. So S_3^4 and S_4^4 can be combined and this merged item can be combined with S_2^4, then $\{1\}$, again resulting in $l(1) = 1$. ∎

This can be shown to be tight by noting that

$$p_\epsilon \triangleq \left(\frac{2\theta}{2\theta+3} - 3\epsilon, \frac{1}{2\theta+3} + \epsilon, \frac{1}{2\theta+3} + \epsilon, \frac{1}{2\theta+3} + \epsilon\right)$$

has optimal length vector $\boldsymbol{l} = (2,2,2,2)$ for any $\epsilon \in (0, (2\theta-1)(8\theta+12)^{-1})$.

Upper bounds derived from this, although rather complicated, are improved.

Corollary 1: For $l(1) = 1$ (and thus for all $p(1) \geq 2\theta(2\theta+3)^{-1}$) and $\theta < 1$, the following holds:

$$\sum_{i=1}^{n} p(i)\theta^{l(i)} > \theta^2 \left[\theta^{\alpha H_\alpha(p)} - p(1)^\alpha\right]^{\frac{1}{\alpha}} + \theta p(1)$$

This is a straightforward consequence of Theorem 1 and a proof is thus omitted for space. This upper bound is tight for $p(1) \geq 0.5$, as $p = (p(1), 1-p(1)+\epsilon, \epsilon)$ gets arbitrarily close for small ϵ.

Let us apply this result to the Benford distribution in (5) for $\theta = 0.6$. In this case, $H_\alpha(p) \approx 2.260$ and $p(1) > 2\theta(2\theta+3)^{-1}$, so $l(1) = 1$ and the probability of success is between 0.251 and $0.315 = \theta^{H_\alpha(p)}$; the simpler (inferior) lower probability bound in (3) is 0.189. The optimal code is $\boldsymbol{l} = (1,2,3,4,5,6,7,8,8)$, which yields a probability of success of 0.296.

VI. Approximation Algorithms and Bounds for Alphabetic Codes

Returning again to alphabetic codes, if the dynamic programming solution is too time- or space-consuming, an approximation algorithm can be used. A simple approximation algorithm involves adding one to each of the lengths of an optimal nonalphabetic code; this yields lengths corresponding to an alphabetic code, since $\sum_i 2^{-l(i)} \leq 0.5$ is sufficient to have an alphabetic code [33, p. 34], [12, p. 565]. Putting the lengths into (2),

$$L_\theta^{\text{huff}}(p) \leq L_\theta^{\text{alpha}}(p) \leq 1 + L_\theta^{\text{huff}}(p)$$

where $L_\theta^{\text{huff}}(p)$ is the cost of the optimal code for the nonalphabetic problem. Limits in terms of Rényi entropy follow from the previous section, and the following improved approximation algorithm means that the right inequality is strict.

Approximation can be improved by utilizing techniques in [12] and [34]. The improved algorithm has two versions, one of which is linear time, using the Shannon-like

$$l^{\S}(i) \triangleq \left\lceil -\alpha \log_2 p(i) + \log_2 \left(\sum_{j=1}^{n} p(j)^\alpha\right)\right\rceil$$

and one of which is $O(n \log n)$ (or linear if sorting weights can be done in linear time).

Procedure for Finding a Near-Optimal Code

1) Start with an optimal or near-optimal nonalphabetic code, $\boldsymbol{l}^{\text{non}}$, such as the Shannon-like $\boldsymbol{l}^{\text{non}} = \boldsymbol{l}^{\S}$ or the Huffman-like $\boldsymbol{l}^{\text{non}} = \boldsymbol{l}^{\text{huff}}$.
2) Find the set of all minimal points, $\mathcal{M}$. A minimal point is any i such that $1 < i < n$, $l(i) < l(i-1)$, and $l(i) < l(i+1)$. Additionally, if $l(i-1) > l(i) = l(i+1) = \cdots = l(i+k) < l(i+k+1)$, then, of these, only $j \in [i, i+k]$ minimizing $w(j)$ (or $p(j)$) is a minimal point.

3) Assign a preliminary alphabetic code with lengths $l^{\text{pre}} = l^{\text{non}}+1$ for all minimal points, and $l^{\text{pre}} = l^{\text{non}}$ for all other items. This corresponds to an alphabetic code C^{pre}. Note that such an alphabetic code is easy to construct; the first codeword is $l(1)$ zeros, and each additional codeword $c(i)$ is obtained by either truncating $c(i-1)$ to $l(i)$ digits and adding 1 to the binary representation (if $l(i) \leq l(i-1)$) or by adding 1 to the binary representation of $c(i-1)$ and appending $l(i) - l(i-1)$ zeros (if $l(i) > l(i-1)$).
4) Go through the code tree (with, e.g., a depth-first search), and replace any node having only one child with its grandchild or grandchildren. At the end of this process, an alphabetic code with $\sum_{i=1}^{n} 2^{-l(i)} = 1$ is obtained.

This hybrid of the approaches of Nakatsu [34] and Yeung [12] can be easily applied to all $\theta > 0$, including the linear limit case, for which it is an improved approximation technique when $\boldsymbol{l}^{\text{non}} = \boldsymbol{l}^{\text{huff}}$.

VII. Related Work, Extensions, and Conclusion

The algorithms presented here will not work if $n = \infty$, although methods are known of finding codes for geometric and lighter distributions [35] and existence results are known for all finite-(Rényi) entropy distributions [36]. Also, although presented here in binary form for simplicity's sake, nonalphabetic results readily extend to D-ary codes [7], [18], [19]. The alphabetic algorithm extends in a manner akin to that shown for the extension of the Gilbert and Moore algorithm in [15, pp. 15-16]. Further upper bounds on optimal $L_\theta(p, \boldsymbol{l})$ are elusive, but should be quite similar to those for the linear case, at least for $\theta < 1$, since the distributions approaching or achieving these bounds should be of bounded cardinality almost everywhere.

In conclusion, when Rényi's siege scenario is formalized, problem solutions involve Huffman coding, dynamic programming, and, appropriately, Rényi entropy.

Acknowledgments

The author would like to thank Thomas M. Cover, T. C. Hu, and J. David Morgenthaler for discussions and encouragement on this topic, as well as the two anonymous reviewers for suggestions on presentation. The author was partially supported, while at Stanford University in the initial phase of this research, by the National Science Foundation (NSF) under Grant CCR-9973134 and the Multidisciplinary University Research Initiative (MURI) under Grant DAAD-19-99-1-0215.

References

[1] A. Rényi, *A Diary on Information Theory*. New York, NY: John Wiley & Sons Inc., 1987, original publication: *Naplò az információelméletről*, Gondolat, Budapest, Hungary, 1976.
[2] P. Mendenhall, "Cell phones were rebels' downfall," Oct. 26, 2002.
[3] J. Taranto, "Best of the Web today," Oct. 28, 2002, available from http://www.opinionjournal.com/best/?id=110002538.
[4] T. Hu, D. Kleitman, and J. Tamaki, "Binary trees optimum under various criteria," *SIAM J. Appl. Math.*, vol. 37, no. 2, pp. 246–256, Apr. 1979.
[5] D. Parker, Jr., "Conditions for optimality of the Huffman algorithm," *SIAM J. Comput.*, vol. 9, no. 3, pp. 470–489, Aug. 1980.
[6] P. Humblet, "Source coding for communication concentrators," Ph.D. dissertation, Massachusetts Institute of Technology, 1978.
[7] ——, "Generalization of Huffman coding to minimize the probability of buffer overflow," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 2, pp. 230–232, Mar. 1981.
[8] G. Markowsky, "Best Huffman trees," *Acta Informatica*, vol. 16, pp. 363–370, 1981.
[9] E. Schwartz, "An optimum encoding with minimum longest code and total number of digits," *Inf. Contr.*, vol. 7, no. 1, pp. 37–44, Mar. 1964.
[10] D. Knuth, "Huffman's algorithm via algebra," *J. Comb. Theory, Ser. A*, vol. 32, pp. 216–224, 1982.
[11] L. Campbell, "Block coding and Rényi's entropy," *Int. J. Math. Stat. Sci.*, vol. 6, no. 1, pp. 41–47, June 1997.
[12] R. Yeung, "Alphabetic codes revisited," *IEEE Trans. Inf. Theory*, vol. IT-37, no. 3, pp. 564–572, May 1991.
[13] E. Gilbert and E. Moore, "Variable-length binary encodings," *Bell Syst. Tech. J.*, vol. 38, pp. 933–967, July 1959.
[14] D. Knuth, "Optimum binary search trees," *Acta Informatica*, vol. 1, pp. 14–25, 1971.
[15] A. Itai, "Optimal alphabetic trees," *SIAM J. Comput.*, vol. 5, no. 1, pp. 9–18, Mar. 1976.
[16] M. Baer, "On conditional branches in search trees," preprint available from http://arxiv.org/abs/cs.PF/0604016.
[17] T. Hu and A. Tucker, "Optimal computer search trees and variable length alphabetic codes," *SIAM J. Appl. Math.*, vol. 21, no. 4, pp. 514–532, Dec. 1971.
[18] L. Campbell, "Definition of entropy by means of a coding problem," *Z. Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 6, pp. 113–118, 1966.
[19] ——, "A coding problem and Rényi's entropy," *Inf. Contr.*, vol. 8, no. 4, pp. 423–429, Aug. 1965.
[20] T. Cover and J. Thomas, *Elements of Information Theory*. New York, NY: Wiley-Interscience, 1991.
[21] C. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, July 1948.
[22] A. Rényi, "On measures of entropy and information," in *Proc. 4th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 1961, pp. 547–561.
[23] S. Newcomb, "Note on the frequency of use of the different digits in natural numbers," *Amer. J. Math.*, vol. 4, no. 1/4, pp. 39–40, 1881.
[24] F. Benford, "The law of anomalous numbers," *Proc. Amer. Phil. Soc.*, vol. 78, no. 4, pp. 551–572, Mar. 1938.
[25] A. Blumer and R. McEliece, "The Rényi redundancy of generalized Huffman codes," *IEEE Trans. Inf. Theory*, vol. IT-34, no. 5, pp. 1242–1249, Sept. 1988.
[26] I. Taneja, "A short note on the redundancy of degree α," *Inf. Sci.*, vol. 39, no. 2, pp. 211–216, Sept. 1986.
[27] R. Gallager, "Variations on a theme by Huffman," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 6, pp. 668–674, Nov. 1978.
[28] O. Johnsen, "On the redundancy of binary Huffman codes," *IEEE Trans. Inf. Theory*, vol. IT-26, no. 2, pp. 220–222, Mar. 1980.
[29] R. Capocelli, R. Giancarlo, and I. J. Taneja, "Bounds on the redundancy of Huffman codes," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 6, pp. 854–857, Nov. 1986.
[30] B. Montgomery and J. Abrahams, "On the redundancy of optimal binary prefix-condition codes for finite and infinite sources," *IEEE Trans. Inf. Theory*, vol. IT-33, no. 1, pp. 156–160, Jan. 1987.
[31] R. Capocelli and A. De Santis, "Tight upper bounds on the redundancy of Huffman codes," *IEEE Trans. Inf. Theory*, vol. IT-35, no. 5, pp. 1084–1091, Sept. 1989.
[32] D. Manstetten, "Tight bounds on the redundancy of Huffman codes," *IEEE Trans. Inf. Theory*, vol. IT-37, no. 1, pp. 144–151, Jan. 1992.
[33] R. Ahlswede and I. Wegener, *Search Problems*. New York, NY: John Wiley & Sons Inc., 1987.
[34] N. Nakatsu, "Bounds on the redundancy of binary alphabetical codes," *IEEE Trans. Inf. Theory*, vol. IT-37, no. 4, pp. 1225–1229, July 1991.
[35] M. Baer, "Integer coding with nonlinear costs," *IEEE Trans. Inf. Theory*, submitted for publication.
[36] ——, "Source coding for quasiarithmetic penalties," *IEEE Trans. Inf. Theory*, submitted for publication.