## IV. Conclusion

In this correspondence, we have presented a powerful technique for generation of linear subcodes of turbo codes with better distance spectra than the mother turbo code, building upon and extending the work of Öberg and Siegel in [2]. We have presented the construction of the linear subcodes as an optimization problem conducted via minimization of a cost function closely related to the upper bound on the asymptotic BER of the code. The minimization itself is achieved via injection of known trace bits at the input of the encoder at strategic locations rendering the occurrence of certain error events impossible, and selective puncturing of the code that allows for recovery of the rate loss incurred in the trace-bit injection process.

Due to the enormous complexity of the direct minimization itself, a greedy approach is adopted that performs iterative optimization by determining the optimal position of the trace bits at a given step, followed by determination of the optimal positions of the punctured bits, without any backtracking (i.e., in a greedy mode). Simulation results are provided validating the effectiveness of the approach by improving the distance spectra of several already optimized PCCC codes.

## References


[1] F. Daneshgaran and M. Mondin, "Design of interleavers for turbo codes: Iterative interleaver growth algorithms of polynomial complexity," *IEEE Trans. Inform. Theory*, vol. 45, Sept. 1999.

[2] M. Öberg and P. H. Siegel, "Application of distance spectrum analysis to turbo code performance improvement," in *Proc. 35th Allerton Conf.*, Urbana-Chamapign, IL, Sept. 1997, pp. 701–710.

[3] S. ten Brink, "Code doping for triggering iterative decoding convergence," in *Proc. IEEE Int. Symp. Information Theory (ISIT2001)*, Washington, DC, June 2001, p. 235.

[4] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. 1995 IEEE Int. Conf. Communications*, Seattle, WA, May 1995, pp. 54–59.

[5] F. Daneshgaran and M. Mondin, "Optimized turbo codes for delay constrained applications," *IEEE Trans. Inform. Theory*, vol. 48, pp. 293–305, Jan. 2002.


# Near-Optimal Depth-Constrained Codes


Pankaj Gupta, *Member, IEEE*, Balaji Prabhakar, *Member, IEEE*, and Stephen Boyd, *Fellow, IEEE*



***Abstract*—This note considers an $n$-letter alphabet in which the $i$th letter is accessed with probability $p_i$. The problem is to design efficient algorithms for constructing near-optimal, depth-constrained Huffman and alphabetic codes. We recast the problem as one of determining a probability vector $q^* = (q_1^*, \ldots, q_n^*)$ in an appropriate convex set, $\mathcal{S}$, so as to minimize the relative entropy $D(p||q)$ over all $q \in \mathcal{S}$. Methods from convex optimization give an explicit solution for $q^*$ in terms of $p$. We show that the Huffman and alphabetic codes so constructed are within 1 and 2 bits of the corresponding optimal depth-constrained codes.**



***Index Terms*—Alphabetic codes, prefix lookup, router.**


## I. Background and Introduction

In the standard binary prefix coding problem, one is required to find binary codewords $c_i$ of lengths $l_i$ for the $n$ letters, $A_1, \ldots, A_n$, of an alphabet such that 1) no codeword is a prefix of another codeword, and 2) if the letter $A_i$ occurs with a probability $p_i$, then the average codeword length $\sum_{i=1}^{n} p_i l_i$ is minimized.

An optimal prefix code can be constructed by Huffman's algorithmic procedure [1]. The Huffman procedure constructs a binary tree with the letters of the alphabet at its leaves, and the codeword $c_i$ represents the path from the root node of the tree to the leaf node associated with the letter $A_i$. The depth of the leaf corresponding to $A_i$, and hence the length of codeword $c_i$, is $l_i$. The depth of the Huffman tree is defined to be the maximum depth of any letter in the tree. The complexity of Huffman's construction is $O(n)$ in both time and space when the probabilities are given in sorted order; otherwise, it is $O(n \log n)$ in time and $O(n)$ in space [2].

If there is an ordering relationship in the alphabet, say $A_i < A_j$ if $i < j$, one is sometimes interested in constructing *alphabetic* codes. Alphabetic codes are prefix codes with the additional restriction that $c_i$ must appear lexicographically before $c_j$ for $i < j$. Equally, in the tree corresponding to an alphabetic code, the letters $\{A_i\}$ must appear in alphabetic order as leaves of the tree. Hu and Tucker [3] and Garsia and Wachs [4] describe procedures for finding the optimal alphabetic code in $O(n \log n)$ time and $O(n)$ space.

An important variant of the general prefix coding problem is the depth-constrained prefix coding problem—the subject of this correspondence. This is the prefix coding problem subject to an additional constraint that the depth of the tree cannot exceed a specified maximum value, say $L$. This problem is useful in many compression and encoding/decoding applications. For instance, it is generally much easier to design efficient decoders when the codes are restricted to a maximum length (such as the word size of the decoding machine).


Manuscript received September 16, 2001; revised July 12, 2004. The work of B. Prabhakar was supported by a Terman Fellowship and an Alfred P. Sloan Fellowship.



P. Gupta is with Cypress Semiconductor, Palo Alto, CA 94301 USA (e-mail: pankaj.gupta@cs.stanford.edu).

B. Prabhakar and S. Boyd are with the Department of Electrical Engineering and Computer Science, Stanford University, Stanford, CA 94305 USA (e-mail: balaji@stanford.edu; boyd@stanford.edu).



Communicated by R. Urbanke, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2004.838345



0018-9448/04$20.00 © 2004 IEEE

Alphabetic trees are used as binary search data structures for conducting queries on $\{A_i\}$. The alphabetic ordering of the leaves allows for simple comparisons at the internal nodes of the tree and guides the search process from the root to a particular leaf. In contrast, a Huffman tree does not necessarily maintain the lexicographic ordering of the letters at its leaves, and hence does not lend itself to efficient binary search procedures. In this correspondence, we are interested in practical and simple-to-implement algorithms for both the Huffman and alphabetic versions of the depth-constrained coding problem.

Our motivation for considering the depth-constrained alphabetic coding problem arose from the need to perform fast route lookups in Internet routers [5]. In the route lookup problem, one is required to search the destination address of every incoming packet to find the best matching entry in the forwarding table of the router. The forwarding table can be organized as a binary search tree. This tree can be redrawn to minimize the average lookup time based on a knowledge of the access probabilities of the different entries. However, it is possible for the depth of the binary tree to grow very large depending on the actual access probabilities. Routers in the core of the Internet have more than 100 000 entries today [6]. This can lead to an alphabetic tree with a worst case depth of several hundreds or thousands, which in turn can cause prohibitively long lookup times (up to 100 000 memory accesses) for some incoming packets. Therefore, it becomes necessary to constrain the depth of the binary tree to some small prespecified number. Moreover, since access patterns to routing table entries can change, ease of implementation of the coding algorithm is more relevant than the strict optimality of the solution. Hence, we are motivated to find *near-optimal* depth-constrained alphabetic codes that can be quickly recomputed.

An algorithm is proposed by Garey [7] for finding the optimal depth-constrained Huffman code in $O(n^2 \log n)$ time, and the optimal depth-constrained alphabetic code in $O(n^3 \log n)$ time. The time complexity of the alphabetic version is improved to $O(n^2 L)$ by Itai [8] and Wessner [9]. Larmore and Hirschberg [10] give an $O(nL)$ time algorithm for the Huffman case, and an $O(nL \log n)$ time algorithm for the alphabetic case. Both these algorithms use a scheme called *Package-Merge*. The fastest optimal algorithm for finding depth-constained Huffman codes is due to Schieber [11], and runs in time $O(n2^{o(\sqrt{\log L \log\log n})})$. Both this and the package-merge algorithms are fairly complicated to implement and have high constant factors. The implementation aspects of the package-merge algorithms are studied by Turpin and Moffat [12].

As we will see, relaxing the optimality restriction affords considerable simplification for finding depth-constrained codes. A near-optimal algorithm was proposed by Mildiú, Laber, and Pessoa [13], [14]. Their algorithm runs in $O(n \log n + n \log w)$[1] time where $w$ is the maximum weight among the letters. The authors prove that the average code length obtained by their algorithm is not greater than the optimal average code length by more than $1/\psi^{L-\lceil \log(n+\log n - L)\rceil - 2}$ where $\psi$ is the golden ratio $\frac{\sqrt{5}+1}{2}$.

This correspondence proposes near-optimal algorithms for the depth-constrained coding problem based on a framework built on convex optimization techniques. The same framework provides solutions for both the Huffman and alphabetic cases. The algorithms are similar in flavor and run in $O(n \log n)$ time and $O(n)$ space. Simple proofs show that the average code lengths obtained by the algorithms are within one and two bits, respectively, of the average codeword lengths of the optimal depth-constrained Huffman and alphabetic codes. The complexity of these algorithms does not depend upon the letter probabilities. More importantly, the algorithms are simple to implement, and the methods used are possibly of general interest.

[1]The bound is stated in terms of integer weights rather than letter probabilities because the algorithm is not strongly polynomial [15].

## II. Depth-Constrained Huffman Codes

We are interested in the following minimization problem: Given a set of probabilities $\{p_i, i = 1, \ldots, n\}$, choose positive integers $\{l_i, i = 1, \ldots, n\}$ so as to minimize

$$C = \sum_{i=1}^{n} l_i p_i \tag{1}$$

subject to the constraints i) $l_i \leq L$ for all $i$, and ii) the $\{l_i\}$ form the lengths of a prefix (or Huffman) code.

Our solution to this problem will use the following two standard lemmas, whose proofs may be found, for example, in Cover and Thomas [16].

*Lemma 1: (Kraft's Inequality):* There exists a prefix code with codeword lengths $\{l_i\}$ if and only if $\sum_{i=1}^{n} 2^{-l_i} \leq 1$.

*Lemma 2:* The minimum average length $C_{\min}$ of a prefix code on $n$ letters, where the $i$th letter occurs with probability $p_i$, satisfies: $H(p) \leq C_{\min} \leq H(p) + 1$, where $H(p)$ is the entropy of the probability distribution $p = \{p_i\}$.

*Remark:* The lower bound, $H(p)$, in Lemma 2 is well known. The upper bound can be achieved using the following codeword lengths (which satisfy Kraft's inequality):

$$l_k = \lceil -\log p_k \rceil, \qquad 1 \leq k \leq n. \tag{2}$$

Since the given set of probabilities $\{p_k\}$ might be such that $p_{\min} = \min_k p_k < 2^{-L}$, the lengths used in the preceding remark could yield a tree whose maximum depth is greater than $L$. To work around this problem we transform the given probabilities $\{p_k\}$ into another set of probabilities $\{q_k\}$ such that $q_{\min} = \min_k q_k \geq 2^{-L}$. With this transformation, we construct a canonical coding tree where the $k$th codeword has a length given by

$$l_k^q = \lceil -\log q_k \rceil, \qquad 1 \leq k \leq n. \tag{3}$$

These lengths satisfy Kraft's inequality and therefore can be used to construct a prefix tree. Clearly, the maximum codeword length is no greater than $L$. We are now left with having to choose a good candidate for the vector $\{q_k\}$ so as to minimize $\sum_k p_k l_k^q$.

Consider

$$\begin{aligned} \sum_k p_k l_k^q &\leq \sum_k p_k \log \frac{1}{q_k} + 1 \\ &= \sum_k p_k \log \frac{p_k}{q_k} - \sum_k p_k \log p_k + 1 \\ &= D(p||q) + H(p) + 1 \end{aligned} \tag{4}$$

where $D(p||q)$ is the relative entropy between the distributions $p$ and $q$.

Therefore, in order to minimize $\sum_k p_k l_k^q$, we must choose $\{q_k\}$ so as to minimize $D(p||q)$. This leads to the following optimization problem: Given $\{p_i\}$ choose $\{q_i\}$ so as to

$$\text{minimize} \quad DPQ = D(p||q) = \sum_i p_i \log(p_i/q_i)$$

$$\text{subject to} \sum_i q_i = 1 \text{ and } q_i \geq Q = 2^{-L}, \qquad \forall i. \tag{5}$$

The cost function $D(p\|q)$ is convex in $(p,q)$ (see [16, p. 30]) and the constraint set is convex; in fact, it is defined by *linear* inequalities. Minimizing convex cost functions with linear constraints is a standard problem in optimization theory and is easily solved by using Lagrange multiplier methods (see, for example, Bertsekas [17, Sec. 3.4] or Boyd and Vandenberghe [18]). Accordingly, define

$$\mathcal{L}(q,\overline{\lambda},\mu)=\sum p_i \log(p_i/q_i)+\sum \lambda_i(Q-q_i)+\mu(\sum q_i-1). \quad (6)$$

Setting the partial derivatives with respect to $q_i$ to zero at $q_i^*$ we get

$$\frac{\partial \mathcal{L}}{\partial q_i}=0 \Rightarrow q_i^*=\frac{p_i}{\mu-\lambda_i}. \quad (7)$$

Substituting this in $\mathcal{L}(q,\overline{\lambda},\mu)$, we get the dual function $\mathcal{G}(\overline{\lambda},\mu)$, and now need to solve the following Lagrange dual problem:

$$\begin{array}{ll} \text{maximize} & \mathcal{G}(\overline{\lambda},\mu) \\ \text{subject to} & \lambda_i \geq 0, \qquad i=1,\ldots,n \end{array}$$

with the implicit constraint that $\mu > \lambda_i$, which is the domain of the dual function $\mathcal{G}$. Now

$$\mathcal{G}(\overline{\lambda},\mu)=\sum_i (p_i \log(\mu-\lambda_i)+\lambda_i Q)+(1-\mu).$$

Maximizing $\mathcal{G}(\overline{\lambda},\mu)$ gives

$$\frac{\partial \mathcal{G}}{\partial \mu}=0 \Rightarrow \sum_i \frac{p_i}{\mu-\lambda_i}=1$$
$$\frac{\partial \mathcal{G}}{\partial \lambda_i}=0 \;\forall i \Rightarrow Q=\frac{p_i}{\mu-\lambda_i}$$

which combined with the constraint that $\lambda_i \geq 0$ gives us $\lambda_i^*=\max(0,\mu-\frac{p_i}{Q})$. Substituting this in (7), we get

$$q_i^*=\max(p_i/\mu, Q). \quad (8)$$

To finish, we need to find a value of $\mu$, $\mu^*$ say, such that the constraint $\sum_{i=1}^n q_i^*=1$ is satisfied. The algorithm MINDPQ described later finds $\mu^*$.

By sorting if necessary we may assume that the $\{p_i\}$ are given to MINDPQ in sorted order with $p_1$ as the largest element and $p_n$ as the smallest element. MINDPQ obtains $\mu^*$ as the solution of $\mathcal{F}(\mu)=0$, where $\mathcal{F}(\mu)=\sum_{i=1}^n q_i^*-1$. For $\mu>0$, let $k_\mu=\inf\{k: \frac{p_k}{Q}>\mu\}$ and $k_\mu=0$ if $\frac{p_k}{Q}\leq\mu$ for all $k$. Using this and (8) we may write $\mathcal{F}(\mu)$ as

$$\mathcal{F}(\mu)=\sum_{i=1}^{k_\mu}\frac{p_i}{\mu}+(n-k_\mu)Q-1. \quad (9)$$

*Lemma 3:* There is a unique value $\mu^*$ of $\mu \in [\frac{p_n}{Q},\frac{p_1}{Q}]$ such that $\mathcal{F}(\mu^*)=0$.

*Proof:* For $0<\mu<\frac{p_n}{Q}$, $k_\mu=n$ and

$$\mathcal{F}(\mu)=\frac{1}{\mu}-1>\frac{Q}{p_n}-1>0$$

since $p_n=\min_k p_k<2^{-L}=Q$. And for $\mu\geq\frac{p_1}{Q}$, $k_\mu=0$ and $\mathcal{F}(\mu)=nQ-1\leq 0$ since $L=\log\frac{1}{Q}\geq\log n$.

Given this, the proof of the lemma will follow from showing that $\mathcal{F}(\mu)$ is a strictly monotonically decreasing function of $\mu$ for

$$\mu\in\left[\frac{p_n}{Q},\frac{p_1}{Q}\right].$$

Accordingly, consider $\mu_1$ and $\mu_2$ such that $\mu_1>\mu_2$. Observe that $k_{\mu_1}\leq k_{\mu_2}$. If $k_{\mu_1}=k_{\mu_2}$, then clearly $\mathcal{F}(\mu_1)<\mathcal{F}(\mu_2)$. If $k_{\mu_1}<k_{\mu_2}$ then

$$\begin{aligned}\mathcal{F}(\mu_1)-\mathcal{F}(\mu_2)&=\sum_{i=1}^{k_{\mu_1}}\frac{p_i}{\mu_1}-\sum_{i=1}^{k_{\mu_2}}\frac{p_i}{\mu_2}+Q\,(k_{\mu_2}-k_{\mu_1})\\&=\sum_{i=1}^{k_{\mu_1}}p_i\left(\frac{1}{\mu_1}-\frac{1}{\mu_2}\right)-\sum_{i=k_{\mu_1}}^{k_{\mu_2}}\frac{p_i}{\mu_2}\\&\quad+Q\,(k_{\mu_2}-k_{\mu_1})\\&<-\sum_{i=k_{\mu_1}+1}^{k_{\mu_2}}\frac{p_i}{\mu_2}+Q\,(k_{\mu_2}-k_{\mu_1})\\&<0\end{aligned}$$

where the last inequality follows from the definition of $k_{\mu_2}$; that is, $\frac{p_i}{\mu_2}>Q$ for all $i\leq k_{\mu_2}$.

The preceding lemma implies that $\mu^*$ can be found explicitly by a binary search procedure. This concludes the description of MINDPQ. Substituting the value of $\mu^*$ in (8) gives us the transformed set of probabilities $\{q_i^*\}$.

*Lemma 4:* Let $l_i^*=\lceil-\log q_i^*\rceil$ for $\{q_i^*\}$ obtained above. Then

$$\sum_i p_i l_i^*\leq H(p)+D(p\|q^*)+1.$$

*Proof:* Follows from (4).

*Theorem 1:* Given an $n$-vector of probabilities $\{p_i\}$, a Huffman code with a depth constraint $L$ can be constructed in $O(n\log n)$ time and $O(n)$ space. The average codeword length so constructed is within one bit of the average length of the optimum depth-constrained Huffman code.

*Proof:* Given the $\{p_i\}$ we execute the algorithm MINDPQ to obtain the associated transformed probabilities $\{q_i^*\}$. The $\{q_i^*\}$ yield codeword lengths $l_i^*$ via (3). These codewords satisfy Kraft's inequality and yield a depth-constrained prefix code. Observe that the MINDPQ procedure which involves sorting takes $O(n\log n)$ time and $O(n)$ space.

Let $\{l_i^{\text{opt}}\}$ be the codeword lengths for the optimum depth-constrained prefix tree with leaf probabilities $\{p_i\}$, and let

$$C_{\text{opt}}=\sum_k p_k l_k^{\text{opt}}=H(p)+D(p\|2^{-l_k^{\text{opt}}})$$

be the associated average codeword length. Now,

$$\begin{aligned}C_{\text{opt}}+1&=H(p)+D(p\|2^{-l_k^{\text{opt}}})+1\\&\overset{\text{(a)}}{\geq}H(p)+D(p\|q^*)+1\\&\geq C_{\min\,dpq}\end{aligned}$$

where (a) follows from the fact that $D(p\|q^*)\leq D(p\|q)$ for all probability distributions $q=\{q_i\}$ such that $\min_i q_i\geq 2^{-L}$.

## III. Depth-Constrained Alphabetic Codes

The algorithm for constructing good depth-constrained alphabetic codes solves the minimization problem stated in (1) of the previous section, with the additional constraint that the resulting code be alphabetic. The solution for the alphabetic version of the problem uses the following two lemmas from Yeung [19], which are analogous to Lemmas 1 and 2, respectively.

*Lemma 5 (Yeung's Characteristic Inequality):* There exists an alphabetic code with codeword lengths $\{l_k\}$ if and only if $s_n \leq 1$, where $s_n$ is given by the recursion

$$\begin{aligned} s_0 &= 0 \\ s_k &= c(s_{k-1}, 2^{-l_k}) + 2^{-l_k} \\ \text{and } c(a,b) &= \left\lceil \frac{a}{b} \right\rceil b. \end{aligned}$$

*Proof:* For a complete proof, see [19]. The basic idea is to construct a *canonical* coding tree, a tree in which the codewords are chosen lexicographically using the lengths $\{l_i\}$. For instance, suppose that $l_i = 4$ for some $i$, and in drawing the canonical tree we find the codeword corresponding to the letter $i$ to be 0010. If $l_{i+1} = 4$, then the codeword for the letter $i+1$ will be chosen to be 0011; if $l_{i+1} = 3$, the codeword for letter $i+1$ is chosen to be 010; and if $l_{i+1} = 5$, the codeword for letter $i+1$ is chosen to be 00110. Clearly, the resulting tree will be alphabetic and Yeung's result verifies that this is possible if and only if the characteristic inequality defined above is satisfied by the lengths $\{l_i\}$.

*Lemma 6:* The minimum average length $C_{\min}$ of an alphabetic code on $n$ letters, where the $i$th letter occurs with probability $p_i$ satisfies

$$H(p) \leq C_{\min} \leq H(p) + 2.$$

*Proof:* The lower bound is well known. For the upper bound, the codeword length $l_k$ of the $k$th letter occurring with probability $p_k$ can be chosen to be

$$l_k = \begin{cases} \lceil -\log p_k \rceil, & k = 1, n \\ \lceil -\log p_k \rceil + 1, & 2 \leq k \leq n-1. \end{cases}$$

The proof in [19] verifies that these lengths satisfy the characteristic inequality, and shows that a canonical coding tree constructed with these lengths has an average depth satisfying the upper bound.

Similar to the Huffman case, since the given set of probabilities $\{p_k\}$ might be such that $p_{\min} = \min_k p_k < 2^{-L}$, a direct application of Lemma 6 could yield a tree whose maximum depth is bigger than $L$. Hence, we again transform the given probabilities $\{p_k\}$ into another set of probabilities $\{q_k\}$ such that $q_{\min} = \min_k q_k \geq 2^{-L}$. The codeword lengths for constructing the canonical coding tree using $\{q_k\}$ are chosen as follows:

$$l_k^q = \begin{cases} \min(\lceil -\log q_k \rceil, L), & k = 1, n \\ \min(\lceil -\log q_k \rceil + 1, L), & 2 \leq k \leq n-1. \end{cases} \tag{10}$$

Each codeword is clearly at most $L$ bits long and the tree thus generated has a maximum depth of $L$. We now show that these codeword lengths yield an alphabetic tree.

*Lemma 7:* The $\{l_k^q\}$ of (10) satisfy the characteristic inequality of Lemma 5.

*Proof:* We shall prove by induction that

$$s_i \leq \sum_{k=1}^{i} q_k, \qquad \forall 1 \leq i \leq n-1.$$

For the base case, $s_1 = 2^{-l_1^q} \leq q_1$ by the definition of $l_1^q$. For the induction step, assume the hypothesis is true for $i-1$. By definition, $s_i = 2^{-l_i^q} + c(s_{i-1}, 2^{-l_i^q})$. Now there are two possible cases.

**Case 1:** $\lceil -\log q_i \rceil + 1 \leq L$. Or, equally, $2^{-(l_i^q - 1)} \leq q_i$. Since

$$c(a,b) \leq \left(\frac{a}{b} + 1\right) b = a + b$$

for all positive $a$ and $b$, we get that

$$s_i \leq 2^{-l_i^q} + s_{i-1} + 2^{-l_i^q}.$$

This and the inductive hypothesis yield

$$s_i \leq 2^{-(l_i^q - 1)} + \sum_{k=1}^{i-1} q_k \leq q_i + \sum_{k=1}^{i-1} q_k = \sum_{k=1}^{i} q_k.$$

**Case 2:** $\lceil -\log q_i \rceil + 1 > L$. This implies that $l_i^q = L$, and hence $q_i \geq 2^{-l_i^q}$. Also, note that the definition of $l_i^q$ at (10) implies, for each $i$, that $l_i^q \leq L$ and hence that $2^{-l_i^q}$ is an integer multiple of $2^{-L}$. This further implies that $s_i$ is an integer multiple of $2^{-L}$ for each $i$. Therefore,

$$c(s_{i-1}, 2^{-l_i^q}) = s_{i-1} \leq \sum_{k=1}^{i-1} q_k$$

and we get

$$s_i = 2^{-l_i^q} + c(s_{i-1}, 2^{-l_i^q}) \leq q_i + \sum_{k=1}^{i-1} q_k = \sum_{k=1}^{i} q_k.$$

Therefore, $s_{n-1} \leq \sum_{i=1}^{n-1} q_i = 1 - q_n \leq 1 - 2^{-l_n^q}$. And

$$\begin{aligned} s_n &= 2^{-l_n} + c(s_{n-1}, 2^{-l_n}) \leq 2^{-l_n^q} + c(1 - 2^{-l_n^q}, 2^{-l_n}) \\ &= 2^{-l_n^q} + 1 - 2^{-l_n^q} = 1. \end{aligned}$$

To continue with our search for good depth-constrained alphabetic codes, we proceed as in (4) and obtain that the lengths defined in (10) satisfy

$$\begin{aligned} \sum_k p_k l_k^q &\leq \sum_k p_k \log \frac{1}{q_k} + 2 \\ &= \sum_k p_k \log \frac{p_k}{q_k} - \sum_k p_k \log p_k + 2 \\ &= D(p\|q) + H(p) + 2 \end{aligned} \tag{11}$$

where $D(p\|q)$ is the relative entropy between the distributions $p$ and $q$.

Thus, given $p$, we need to determine

$$q^* \in \left\{ q : \min_i q_i \geq 2^{-L}; \sum_i q_i = 1 \right\}$$

so that $D(p\|q)$ is minimized. This convex optmization problem can be solved exactly analogously as before using the algorithm MINDPQ. The $q^*$ so determined will yield codeword lengths $\{l_i^*, 1 \leq i \leq n\}$ via (10). And, as shown in (11), the average codeword length is within two bits of that of the optimum depth-constrained alphabetic code.

The above results are stated in the following theorem whose proof is identical to that of Theorem 1, and hence is omitted.

*Theorem 2:* Given an $n$-vector of probabilities $\{p_i\}$ an alphabetic code with a depth constraint of $L$ can be constructed in $O(n \log n)$ time and $O(n)$ space. The average codeword length so constructed is within two bits of the average length of the optimum depth-constrained alphabetic code.

## REFERENCES


[1] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, pp. 1098–1101, Sept. 1952.

[2] J. van Leeuwen, "On the construction of Huffman trees," in *Proc. 3rd Int. Colloquium on Automata, Languages and Programming*, Edinburgh, U.K., 1976, pp. 382–410.

[3] T. C. Hu and A. C. Tucker, "Optimal computer search trees and variable length alphabetic codes," *SIAM J. Appl. Math.*, vol. 21, pp. 514–532, 1971.

[4] A. M. Garsia and M. L. Waschs, "A new algorithm for minimal binary search trees," *SIAM J. Comput.*, vol. 6, pp. 622–642, 1977.

[5] P. Gupta, B. Prabhakar, and S. Boyd, "Near-optimal routing lookups with bounded worst case performance," in *Proc. INFOCOM*, vol. 3, Tel-Aviv, Israel, Mar. 2000, pp. 1184–1192.
[6] Bgp Table Statistics [Online]. Available: http://bgp.potaroo.net/
[7] M. R. Garey, "Optimal binary search trees with restricted maximal depth," *SIAM J. Comput.*, vol. 3, pp. 622–642, 1974.
[8] A. Itai, "Optimal alphabetic trees," *SIAM J. Comput.*, vol. 5, pp. 9–18, 1976.
[9] R. L. Wessner, "Optimal alphabetic search trees with restricted maximal height," *Inform. Processing Lett.*, vol. 4, pp. 90–94, 1976.
[10] L. L. Larmore and D. S. Hirschberg, "A fast algorithm for optimal length-limited Huffman codes," *J. ACM*, vol. 37, pp. 464–473, 1990.
[11] B. Schieber, "Computing a minimum-weight k-link path in graphs with the concave monge property," in *Proc. 6th ACM IEEE Symp. Discrete Algorithms*, Jan. 1995, pp. 405–411.
[12] A. Turpin and A. Moffat, "Practical length limited coding for large alphabets," *Computer J.*, vol. 38, no. 5, pp. 339–347, 1995.
[13] R. L. Milidiú, A. A. Pessoa, and E. S. Laber, "Efficient implementation of the WARM-UP algorithm for the construction of length-restricted prefix codes," in *Proc. ALENEX (Lecture Notes in Computer Science)*. Berlin, Germany: Springer-Verlag, 1999, vol. 1619.
[14] R. L. Milidiú and E. S. Laber, "Warm-Up Algorithm: A Lagrangean Construction of Length Restricted Huffman Codes," Dep. Informática, PUC-RJ, Rio de Janeiro, Brasil, Tech. Rep. 15, 1996.
[15] E. S. Laber, private communication.
[16] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1995, Series in Telecommunications.
[17] D. P. Bertsekas, *Nonlinear Programming*. Nashua, NH: Athena Scientific, 1995.
[18] S. Boyd and L. Vandenberghe. (2001) Convex Optimization (Course reader for EE364, "Introduction to Convex Optimization with Engineering Applications"). [Online]. Available: http://www.stanford.edu/class/ee364/
[19] R. W. Yeung, "Alphabetic codes revisited," *IEEE Trans. Inform. Theory*, vol. 37, pp. 564–572, May 1991.

# Codes From the Suzuki Function Field

Gretchen L. Matthews, *Member, IEEE*

***Abstract*—We construct algebraic geometry (AG) codes from the function field $\mathbb{F}_{2^{2n+1}}(x,y)/\mathbb{F}_{2^{2n+1}}$ defined by**

$$y^{2^{2n+1}} - y = x^{2^n}(x^{2^{2n+1}} - x)$$

**where $n$ is a positive integer. These codes are supported by two places, and many have parameters that are better than those of any comparable code supported by one place of the same function field. To define such codes, we determine and exploit the structure of the Weierstrass gap set of an arbitrary pair of rational places of $\mathbb{F}_{2^{2n+1}}(x,y)/\mathbb{F}_{2^{2n+1}}$. Moreover, we find some codes over $\mathbb{F}_8$ with parameters that are better than any known code.**

***Index Terms*—Algebraic geometry (AG) code, optimal function field, Suzuki curve, Suzuki function field, Weierstrass gap set.**

## I. Introduction

IN [3], the function field $\mathbb{F}_8(x,y)/\mathbb{F}_8$ defined by

$$y^8 - y = x^2(x^8 - x)$$

is used to construct codes supported by a single place that have better parameters than any known code. Such codes are sometimes referred to as one-point codes. In [14], it is shown that there are $m$-point codes with $m \geq 2$, that is algebraic geometry (AG) codes supported by $m$ places where $m \geq 2$, that have better parameters than any comparable one-point code constructed from the same curve. In this correspondence, we combine these ideas to find such two-point codes over $\mathbb{F}_{2^{2n+1}}$ where $n$ is a positive integer. Of those we find, some have better parameters than any comparable one-point code and some have better parameters than any known code.

We consider the function field $F := \mathbb{F}_q(x,y)/\mathbb{F}_q$ defined by

$$y^q - y = x^{q_0}(x^q - x)$$

where $q_0 = 2^n$, $q = 2^{2n+1}$, and $n$ is a positive integer. The projective curve $X$ defined by the above equation was considered in [8] as an example of a curve with an automorphism group that is large with respect to its genus. The curve $X$ (resp., function field $F$) is sometimes called the Suzuki curve (resp., Suzuki function field) as the automorphism group of $X$ (resp., $F$) is the Suzuki group of order $q^2(q^2+1)(q-1)$. In [10], Hansen and Stichtenoth considered this curve and applications to AG codes. More recently, Kirfel and Pellikaan determined the Feng-Rao bound on the minimum distances of some of these AG codes [12]. The case where $n = 1$ has been examined by Chen and Duursma as mentioned above. Here, we use the structure of Weierstrass gap sets to construct codes and estimate their parameters. This method was first suggested by Goppa ([6], [7]) and later made more explicit in [5], [14], [9], [4], and [13]. We see that these new codes compare quite favorably with those studied in [10], [12], and [3].

This work is organized as follows. Section II contains the necessary background information on the Suzuki function field. In Section III, we determine the Weierstrass gap set of any pair of rational places. In Section IV, this gap set is used to define two-point AG codes. These codes are compared with one-point codes constructed from the Suzuki function field. In addition, we find codes over $\mathbb{F}_8$ other than those in [3] with better parameters than any known code.

## II. Suzuki Function Fields

Let $F := \mathbb{F}_q(x,y)/\mathbb{F}_q$ denote the algebraic function field defined by

$$y^q - y = x^{q_0}(x^q - x)$$

where $q_0 = 2^n$ and $q = 2^{2n+1}$ for some positive integer $n$. Let us review some facts about $F/\mathbb{F}_q$ found in [10]. The notation we use is as in [16]. A place of $F/\mathbb{F}_q$ of degree one will be called a rational place. The set of all rational places of $F/\mathbb{F}_q$ is denoted by $\mathbb{P}_F$, and the divisor (resp., pole divisor) of a function $f \in F$ is denoted by $(f)$ (resp., $(f)_\infty$). The function field $F/\mathbb{F}_q$ has exactly $q^2+1$ rational places. In fact, for each $a, b \in \mathbb{F}_q$ there exists a unique rational place $P_{ab} \in \mathbb{P}_F$ that is a common zero of $x-a$ and $y-b$. In addition, $F$ has a single place at infinity, $P_\infty$. The genus of $F$ is $g := q_0(q-1)$. Moreover, the explicit formulas of Weil can be used to show that $F$ is an optimal function field.

It will be convenient at times to view $F$ as an extension of the rational function field $\mathbb{F}_q(x)$. Then $[F : \mathbb{F}_q(x)] = q$ (see [10, Lemma 1.8]). Let $Q_a \in \mathbb{P}_{\mathbb{F}_q(x)}$ denote the zero of $x-a$ and $Q_\infty \in \mathbb{P}_{\mathbb{F}_q(x)}$ denote the place at infinity. It will also be useful to consider the functions

$$x, y, v := y^{\frac{q}{q_0}} - x^{\frac{q}{q_0}+1}, \quad w := y^{\frac{q}{q_0}} x^{\frac{q}{q_0^2}-1} + v^{\frac{q}{q_0}} \in F$$

Manuscript received September 10, 2003; revised March 19, 2004. This work was supported by the National Science Foundation under Grant DMS-0201286.

The author is with the Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-0975 USA (e-mail: gmatthe@clemson.edu).

Communicated by K. A. S. Abdel-Ghaffar, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2004.838102

0018-9448/04$20.00 © 2004 IEEE