Code and Parse Trees for Lossless Source Encoding

Julia Abrahams
Mathematical, Computer, and Information Sciences Division
Office of Naval Research
Arlington, VA 22217-5660
abrahaj@onr.navy.mil

I. Introduction

This paper surveys the theoretical literature on fixed-to-variable-length lossless source code trees, called code trees, and on variable-length-to-fixed lossless source code trees, called parse trees. Huffman coding [1] is the most well known code tree problem, but there are a number of interesting variants of the problem formulation which lead to other combinatorial optimization problems. Huffman coding as an instance of combinatorial search has been highlighted in the books by Ahlswede and Wegener [2] and Aigner [3]. See also the papers of Hinderer and Stieglitz [4] and Hassin and Henig [5] for overviews of the combinatorial search literature. Tunstall parsing [6] is the most well known parse tree problem for a probability-based source model, although parsing based directly on source data is very familiar as Lempel Ziv parsing [7-8], a family of techniques which is outside the scope of this survey. Similarly, adaptive, data-based variants of Huffman coding, e.g. [9-12] will not be treated here. Rather, the assumption here is that the source model is given as a sequence of independent and identically distributed (iid) random variables for some known discrete distribution, although on occasion it is possible that only partial information about the source is available. These lossless source encoding techniques comprise a subset of data compression techniques, and broader surveys of the data compression literature are available [13-21].

In particular, the following code tree topics are outlined in this survey: characteristics of the Huffman code tree; Huffman-type coding for infinite source alphabets and universal coding; the Huffman problem subject to a lexicographic constraint, or, that is, the Hu-Tucker problem; the Huffman problem subject to maximum codeword length constraints; code trees which minimize other functions besides average codeword length; coding for unequal cost code symbols, or, that is, the Karp problem, and finite state channels; and variants of Huffman coding in which the assignment of 0's and 1's within codewords is significant such as bidirectionality and synchronization. The literature on parse tree topics is less extensive. Treated here are: variants of Tunstall parsing; dualities between parsing and coding; dual tree coding in which parsing and coding are combined to yield variable-length-to-variable-length codes; and parsing and random number generation. Finally, questions related to counting and representing code and parse trees are also discussed.

II. Code Trees

In fixed-to-variable-length binary source coding, a code tree is used to associate a string of binary code symbols with each of a set of K source symbols. A code tree is a complete binary tree with K leaf nodes and K-1 internal nodes including the root at the top of the tree. Pairs of branches descend from each internal node. Left branches are labeled with the code symbol 0 and right branches are labeled with the code symbol 1. Each leaf node is labeled with one of the K source symbols. A path through the code tree from the root to a leaf describes the string of code symbols associated with the source symbol at the leaf. The problem is to find a code tree to optimize some performance criterion, possibly under the imposition of constraints on the form of the tree. For r-ary source coding, when r>2, each internal node of the tree can have anywhere between 2 and r branches descending from it, and this flexibility introduces

0-8186-8132-2/98 $10.00

some additional complexity to the r-ary code tree problem. Nevertheless, much of the literature concentrates on binary code trees with the r-ary generalization straightforward.

II. 1. Huffman Code Trees

Huffman coding [1] can be found described in all information theory textbooks [22-25], as well as in many computer science [26-30] and discrete mathematics texts [31-32]. Huffman's algorithm solves the quintessential problem of finding the code tree to minimize average codeword length. For source symbols occurring with probabilities $P=\{p(i), i=1, ...,K\}$ we wish to find path lengths through the tree $L=\{l(i), i=1, . . .,K\}$ to solve $\min_L \Sigma_{1\le i\le K} p(i)l(i)$. There are no further constraints on the problem; however, it is known from the Kraft inequality that for a set of codeword lengths L to be compatible with a binary code tree it is necessary and sufficient that $\Sigma 2^{-l(i)} \le 1$ hold. Because the resulting minimum average codeword length l^* can be bounded in terms of entropy $H=-\Sigma p(i)\log_2 p(i)$ according to $H \le l^* < H+1$ with $l^*=H$ if and only if P is a dyadic distribution, that is $p(i)= 2^{-l(i)}$ for some integer l(i), minimum redundancy, or l^*-H, is bounded according to $0 \le l^*-H < 1$. An equivalent perspective on Huffman coding is that it minimizes redundancy over L because H is a constant with respect to the minimization. Another equivalent perspective on Huffman coding is that the dyadic distribution induced by the code tree $Q=\{q(i)=2^{-l(i)}, i=1, ...,K\}$ is the dyadic distribution closest to P in the minimum discrimination sense. That is to say, Huffman coding solves $\min_L \Sigma p(i)\log_2 p(i)/q(i)$.

Huffman's algorithm is a bottom up merge type algorithm in which, at each stage of the process, the two least probable source symbols are "merged" into a pair of sibling nodes whose new parent node has probability given by the sum of the two original probabilities. These combined nodes are treated as a single node in each subsequent stage of the algorithm. Note that the code tree generated by the Huffman algorithm is not unique, and that there are code trees which share the same set of codeword lengths as a Huffman code but which cannot be obtained from the Huffman algorithm. These distinctions will not be important here.

II.2. Characterizations of Huffman Code Trees

A number of authors have addressed the problem of characterizing the form of the resulting Huffman code tree given partial or full information about the source probabilities. These are results about the form of the Huffman code tree which are known without actually constructing the code. Katona and Nemetz [33] upper bound the length of the codeword associated with a source symbol of probability p(i) based on the extremal properties of the Huffman code tree for a particular source distribution involving ratios of Fibonacci numbers. They are motivated to discover a relationship between the self-information of the source symbol, $-\log_2 p(i)$, and its codeword length, l(i), in view of the entropy bounds on average codeword length. Other work on individual Huffman codeword lengths has also appeared [34-41]. Schack [42] continues Katona and Nemetz's line of inquiry by bounding the probability of source symbols whose self-information and codeword lengths are far from each other. Cheng et al. [43] make use of Schack's machinery in their work on bounding the temporary expansion possible when Huffman coding is used in lieu of block coding and low probability source symbols precede high probability source symbols in encoding a sequence of source symbols. De Prisco and De Santis [44] further address the data expansion problem by making use of the literature on redundancy bounds for given partial information about the source probabilities.

When the given source symbol is known to be the most likely symbol, with probability, say, p(1), its codeword length can be specified to within 1 [34-35, 45]. Results of this type have typically been employed to bound the redundancy of the Huffman code in terms of known p(1) more tightly than to within [H, H+1). The most comprehensive paper on redundancy bounds is Manstetten's [46]. The

problem of redundancy bounds originated with Gallager [9], and a number of other authors contributed to this problem [47-53] including the problem variant in which the least likely probability, p(K), is known. A useful general approach to establishing redundancy bounds given a subset of the ordered source probabilities is Yeung's [54] redundancy theorem, and several of the redundancy bounds are special cases of his result.

Golomb [55] considers finite source distributions for which multiple distinct code trees share a common minimum average codeword length.

Geçkinli [56] gives necessary and sufficient conditions on the source probabilities P for the difference between the longest and shortest codeword lengths to be 0 or 1. Katona and Lee [34] give a sufficient condition such that the difference is less than or equal to a given integer. In a sense, Geçkinli addresses the question of, when is the short, fat tree optimal? In contrast, Katona and Nemetz's [33] work can be interpreted as addressing the question of when is the long, thin tree (with codeword lengths {l(i)=i, i=1,..., K-1; l(K)=l(K-1)})optimal? Vinokur [57] addressed this question independently as well.

Some authors are concerned with the balance properties of Huffman trees, that is, with the relative size of the weights associated with the internal nodes of the tree [58-60].

For some parametric families of source distributions, the form of the resulting Huffman code tree is known explicitly. The binomial distribution arises in encoding fixed length blocks of binary source symbols from an independent and identically distributed sequence of biased coin tosses. Jakobsson [61] gives the Huffman code in explicit form for binomial parameters satisfying certain conditions. Stubley [62] considers the binomially distributed source for all parameter values and analyzes the redundancy of the Huffman code in this case obtaining redundancy bounds. Stubley [63] also examines the multinomial distribution arising from fixed length blocks of r-ary source symbols. Other authors who have examined binomial sources have been interested in questions about the behavior of the resulting average codeword length as a function of block size, and in particular about its nonmonotonicity with increasing block size. See [45, 64-66].

Similar questions related to Huffman coding variable length strings rather than fixed length blocks can be thought of as arising within the framework of variable-to-variable-length or dual tree coding, treated in a later section.

Another distribution which has been studied in connection with Huffman coding is the finite geometric distribution. Huffman coding for this distribution arises in the context of group testing, a combinatorial search problem in which the goal is to fully classify all members of a set into one of two classes based on a series of tests, each of which can determine whether or not the subset submitted to it consists entirely of members of one of the classes. Hwang [67] finds the average codeword length for this distribution without explicitly identifying the codeword lengths. The codeword lengths are given explicitly for some parameter values only in [68]. Other work on this problem appears in [4, 69].

There are also a few other parametric families of finite source distributions for which Huffman codes are available. The uniform distribution always leads to a short, fat Huffman tree as in Geçkinli [56]. Günther and Schneider [70] and Campbell [71] discuss the "typical" source of K symbols and compare its entropy to that of the uniform source. The difference is quite small for large K suggesting that, for large, typical sources, the short, fat tree is nearly optimal. And of course the Huffman code trees for dyadic distributions are immediate by inspection. These Huffman codes have zero redundancy and the interesting property that 0's and 1's occur with equal probability on average in the string of code symbols, a property which does not hold in general [72]. The finite zeta-function distribution in which p(i) is proportional to 1/i is studied by Gutman [73] who shows that for certain finite source alphabet sizes the Huffman code tree can be found explicitly for this distribution. Tucker [74] examines probabilities proportional to i, i=1,...,K. Hwang [75] considers the case that the probabilities take on either of only two

values. A distribution of particular form arising in a data structure application is explicitly Huffman coded in [76]. There are also a number of infinite source distributions for which explicit minimum average length codes are known, and these will be treated in the next section.

Gallager's sibling property [9] states that a code tree is a Huffman tree if and only if the nodes of the tree can be listed in order of nonincreasing probability with each node being adjacent in the list to its sibling. This property can be used to verify whether or not a given tree is the Huffman tree for a fixed distribution.

Longo and Galasso [77] employ the minimum discrimination perspective for Huffman coding and find conditions on source probability distributions such that they share the same Huffman code as a particular dyadic distribution. See also [78]. Discrimination arises also in the problem of evaluating the mismatch when a source with one set of probabilities is mistakenly encoded assuming a different set of probabilities [79-81]. It may be of interest to examine mismatch problems in other coding and parsing contexts as well.

Hu and Tucker [82] and Hwang [83] consider the question of computing the average codeword lengths of Huffman codes for different source probability distributions without actually constructing the codes. They give inequalities on functionals of the source probabilities sufficient for inequalities on the Huffman average codeword lengths. Their approaches are related to the notion of majorization in the theory of inequalities.

Hwang's [83] paper is interesting in another respect. He deals with Huffman code trees more general than r-ary. The number of branches descending from each internal node is a fixed integer varying from node to node. See also Chu and Gill [84] for trees with variable internal node degrees for the Huffman problem. Both Hwang and Chu and Gill also address the Hu-Tucker problem discussed later.

II. 3. Infinite Source Alphabets and Universal Coding

Since the Huffman algorithm is bottom up, beginning with the merge of the two least likely source symbols, infinite source alphabets are not immediately amenable to Huffman coding. However, Gallager and Van Voorhis [85] were able to make use of the Huffman code for a finite distribution obtained from the infinite distribution together with a limiting argument and obtain the minimum average codeword length code for the infinite geometric distribution. Their work extends the earlier work of Golomb [86] on this problem. Note that the geometric distribution arises in describing the run lengths generated by a sequence of underlying iid binary random variables. Similar methods have been applied to other parametric families of infinite source distributions or to establishing sufficient conditions on the source probabilities for a particular structured infinite tree to be optimal for that distribution [87-90]. Linder et al. [91] prove that this general approach will find the optimal code whenever the source entropy is finite, however it is not truly a constructive method. Kato [92] has recently extended the proof to infinite entropy distributions. Several authors, for example [93-98], use the codeword sets of Golomb and Gallager and Van Voorhis or a modification of these sets for other infinite source distributions besides geometric, not necessarily in an optimal fashion.

Gallager and Van Voorhis's infinite code trees, parameterized by an integer which depends on the parameter of the geometric distribution, are not universal in the sense of Elias [99]. Universal codes for infinite alphabets have the property that when the shortest codewords are used to represent the most likely source symbols, the resulting average codeword length is upper bounded by a constant multiple of the source entropy for all infinite source distributions.

A number of authors discuss infinite code trees in the context of universal codes including [100-111]. Many of the universal code trees discussed in the literature fall into a common framework as described in [18, Sect. 3.3] based on systems of numeration [112]. Some specific examples of universal

codes based on numeration systems are identified in [113-115] and independently in [116]. The general idea is that each positive integer x is assigned a binary codeword composed of two components consistent with the existence of a binary code tree. An infinite sequence $V=\{V_i, i=1,2,\ldots\}$ is specified and x is represented relative to V by the pair (j,r) where $\Sigma_{1\le i\le j} v_i < x \le \Sigma_{1\le i\le j+1} v_i$ and $r=x-\Sigma_{1\le i\le j} v_i - 1$. The concatenation of a string of j bits for j and a string of $\lceil \log v_j \rceil$ bits for r makes up the codeword. Of course nonuniversal infinite alphabet code trees can also be interpreted in terms of the two components of a numeration system, particularly when the convention for the representation of r is loosened to include other variable length strings. There are a variety of other approaches to universal coding which do not fall into the infinite code tree perspective and are not discussed here.

II. 4. Lexicographic Constraints on the Huffman Coding Problem: The Hu-Tucker Problem

Suppose that the source symbols to be encoded must be assigned to the leaves of the code tree in a fixed left-to-right order. Equivalently the binary codewords must exhibit a particular lexicographic order. It may happen for a particular set of source probabilities and a particular linear order constraint on the corresponding symbols that the Huffman code tree (or a tree with the same average codeword length in the case that the minimum average codeword length tree is not unique) exhibits the desired linear order. This is the case, for example, if the source symbols are linearly ordered according to a monotonically increasing or decreasing order on their probabilities. But, in general, the Huffman code tree does not exhibit the desired linear order, and the code tree which minimizes average codeword length subject to a linear order constraint on the source symbols has a resulting average codeword length greater than the Huffman minimum. Such alphabetic or lexicographic or linearly ordered code trees can be obtained using the algorithm of Hu and Tucker [117-121], and their average codeword lengths fall within [H, H+2). The original work on alphabetic codes goes back to Gilbert and Moore [122].

The Hu-Tucker algorithm has as its central step a merge operation like the Huffman merge but restricted to pairs of symbols which are adjacent in the linear order in a certain generalized sense. Repeated application of this merge operation yields a tree which is then able to be rearranged into a tree with the same set of codeword lengths which also satisfies the linear order constraint. It is a difficult algorithm to understand in contrast with the straightforwardness of the Huffman algorithm.

Several authors have proved a version of the Kraft inequality for Hu-Tucker code trees, that is, have characterized the set of codeword lengths consistent with the existence of a lexicographically constrained tree [123-126]. With this result in hand, a number of results refining the [0,2) Hu-Tucker code redundancy bound given partial information about the source probabilities and their linear order have been obtained, along with bounds on the Hu-Tucker average codeword length given in terms of the Huffman average codeword length for the same (unordered) probabilities [123-126]. Kleitman and Saks's [127] result, that the worst case ordering of source probabilities for the Hu-Tucker problem is the "sawtooth order" also serves to upper bound the Hu-Tucker average codeword length. Hu and Tan [128] take this kind of approach to performance bounds in a more general ordered search tree context which includes the Hu-Tucker problem as a special case.

One other set of results characterizing the form of the Hu-Tucker code tree is due to Ramanan [129] who gives inequalities on the source probabilities sufficient for a particular Hu-Tucker tree to be optimal. Ramanan's motivation is in finding means to test whether a given tree is the Hu-Tucker tree for a fixed set of source probabilities in a way which is computationally less complex than the construction of the optimal tree itself. This may be a potentially promising perspective to take, as well, towards other constrained coding problems, to be described later in this survey, for which the optimal code tree is difficult to find algorithmically.

A stricter set of constraints on the Huffman problem than the linear order of Hu and Tucker is Van Voorhis's [130] problem in which the codewords must in addition satisfy a monotonic length constraint. As expected, the resulting minimum average codeword length code obtained from Van Voorhis's algorithm has an average codeword length which can be upper bounded in terms of the Hu-Tucker codeword length [131].

One combinatorial search problem besides the minimum average codeword length coding problem in which Huffman and Hu-Tucker code trees arise is the problem of testing for a single distinguished item on the basis of a series of tests each of which can identify whether the distinguished item is in a given subset of items presented to the test or not. We are given prior probabilities on each item as to how likely it is to be the distinguished item. If the items are components of some system arranged in a fixed left-to-right order such as in an "oil pipeline", and the tests identify whether or not the distinguished or "faulty" component is to the left of the tested point, the test tree which minimizes the average number of tests to identify the distinguished component is the Hu-Tucker tree. The problem variant in which the components are arranged according to a fully-connected, partially-ordered structure with a single final element in the partial order has been examined [132] but not fully resolved. It would be of interest to have a generalization of the Hu-Tucker algorithm which applied to a partial-order constraint of this form on the source symbols. Similarly one can imagine imposing partial orders of other forms on the source symbols in a minimum average codeword length coding problem, for example, consider a set of several disjoint linear orders, representing a source alphabet of letters, numbers, and punctuation marks, and an algorithm and bounds on the resulting codeword length would be of interest in this case as well. For constrained trees in these and other search problems, the appropriate Kraft-type inequality would be of interest although in most cases it is not available.

For a parallel search of linearly ordered items, or for a parallel version of the unordered Huffman problem, an optimal forest is found by stopping the Hu-Tucker or Huffman merge algorithms after K-m merge steps [117]. Then each of the m trees in the forest can be searched simultaneously, and the average time for any of the searchers to find the single distinguished time is minimized. Entropy bounds on the resulting minimum average path lengths are given in [133] for these problems.

Another potentially interesting variant of the Hu-Tucker problem may be the "parallel product" problem described in [4]. Imagine tests of the Hu-Tucker type used to identify a single distinguished item, however each item is characterized by D>1 attributes, each one of which can only be tested separately. For example, the single distinguished item may be thought of as faulty, but to be faulty it must be defective in all of D dependent but separately testable modes. We are given the prior probability that the item for which attribute d takes on the value i(d), i(d)=1,. . ., K(d), d=1,. . .,D, is the distinguished item. Thus each node in the code or test tree distinguishes between the subset of the items for which i(d) is less than some value and its complement. In essence this problem introduces probabilities into the standard multidimensional search problem in which the attributes are tested cyclically, d=1, ...,D, in each level of the tree. A Huffman "parallel product" problem or one which is Huffman in some attributes and Hu-Tucker in the others might also be interesting.

In any of these Huffman problem variants with order constraints, the average codeword length obtained from the Huffman algorithm applied to the unconstrained probabilities is always a lower bound on the constrained minimum average codeword length. If the Huffman tree satisfies the constraints of the problem, or if it can be rearranged into a tree with the same codeword lengths which satisfies the constraints of the problem, then that tree solves the constrained problem. The characterization of constraint "profiles" for which the unconstrained, Huffman, algorithm applies may be useful.

Note that the nonbinary Hu-Tucker problem remains essentially open [134].

II.5. Maximum Codeword Length Constraints on Huffman and Hu-Tucker Coding

Both the Huffman and Hu-Tucker problems have been addressed under the additional constraint that no codeword be longer than a certain maximum value. Clearly, as that permitted maximum value increases, eventually the unconstrained Huffman or Hu-Tucker tree results. Also, in general, the imposition of the maximum codeword length constraint will lead to larger resulting average codeword lengths than in the unconstrained Huffman and Hu-Tucker problems.

Currently the best algorithmic approach to Huffman coding under the maximum codeword length constraint is due to Moffat et al. [135-137] based on the algorithm of Larmore and Hirschberg [138]. However, a number of earlier approaches to the same problem have appeared [139-146]. And see [147] for a Hu-Tucker coding algorithm under the maximum codeword length constraint.

Some performance analysis results are available for maximum codeword length constrained coding problems, notably Capocelli and De Santis's [148] redundancy bounds for the length constrained Huffman problem, including the case of p(1) or p(K) given. The corresponding questions for the length constrained Hu-Tucker problem have not been addressed. The worst case ordering of source probabilities for the length constrained Hu-Tucker problem remains the "sawtooth order" as in the unconstrained problem [149].

The work of Anily and Hassin [150] is relevant to constraints such as maximum length constraints or other constrained problems which one might propose, such as maximum constraints on the difference between longest and shortest codeword lengths, or fringe, for example. They compute the best, second best,. . .,s^{th}-best trees in terms of minimum average codeword length for fixed source probabilities from which the best tree satisfying the constraints can be identified by examining each candidate tree in turn. They deal with both the Huffman and Hu-Tucker versions of this problem. Entropy bounds on the average codeword length of the s^{th}-best tree are not available and may be of interest to obtain.

II.6. Minimizing Other Codeword Length Functionals

Both the Huffman and Hu-Tucker problems have been addressed under the variant that a functional of codeword length other than average codeword length is to be minimized. Of particular interest are cases in which a merge type algorithm like Huffman or Hu-Tucker serves to construct the optimal tree. Parker [151] has unified a large earlier literature on generalized Huffman problems into a common comprehensive framework. In particular, Parker works with a class of a "quasilinear" merge functions, F(x,y), whose arguments are the probabilities weighting the nodes to be merged. Then F(x,y) will be the weight of the newly created parent node. In the ordinary Huffman algorithm, $F_s(x,y)=x+y$. Another example of merge function which is included in Parker's framework is $F_m(x,y)=\max(x,y)+c$. Then Parker's generalized merge algorithm creates a tree by merging, at each stage of the process, the two least probable source symbols, with probabilities x and y, into a pair of sibling nodes whose new parent node has probability, or weight, given by F(x,y). If G is a "Schur concave" cost functional of the weights on the interior nodes of the tree, then the generalized merge algorithm forms a tree for which G is minimized. In the ordinary Huffman algorithm, G_s=sum, and the resulting tree which minimizes the sum of the weights of the interior nodes of the tree is exactly the tree which minimizes average codeword length $\Sigma p(i)l(i)$.

Another example of cost functional which is included in Parker's framework is G_m=max, and the tree resulting from the generalized merge algorithm incorporating F_m which minimizes the maximum of the weights of the interior nodes of the tree is exactly the tree which minimizes max (p(i)+cl(i)). Generalized entropy-type bounds on the resulting minimum codeword length functional are included in [151]. See also [152] for p(1) given. Among the special cases included in Parker's framework are those addressed in [153-155] and see the references in [151]. Knuth [156] revisited Parker's problem from an

elegant abstract perspective which also incorporates Huffman problems with secondary optimality criteria such as, of all Huffman trees for given source probabilities, select the one with minimum sum of codeword lengths, a problem originated by Schwartz [157], or minimum variance, a problem originated by Kou [158]. See [159] for more on the Schwartz and Kou problems. Markowsky [160] also examines Huffman problems with secondary optimality criteria. Chang and Thomas [161] further examined Knuth's algebraic perspective on code trees, particularly in the case that random variables are substituted for the source probabilities.

The generalized merge version of the Hu-Tucker problem was addressed by Hu, Kleitman, and Tamaki [134] who work with a class of merge functions and cost functionals satisfying a set of specified properties. It is not immediately clear exactly what is the correspondence between the class of F,G pairs for which Parker's generalized Huffman algorithm holds and the class of F,G pairs for which Hu, Kleitman, and Tamaki's generalized Hu-Tucker algorithm holds. It may be a useful contribution to align these two apparently independent generalized merge formalisms. It may also be of interest to obtain generalized entropy-type bounds on the minimum codeword length functional for alphabetic codes in the context of the Hu, Kleitman and Tamaki algorithm. Zhang [162] proves that a generalized merge Hu-Tucker type algorithm finds the optimal tree for a particular F,G pair, F_m for c=1 and G_s, which is claimed to be outside the original class of F and G treated by Hu, Kleitman, and Tamaki. Kirkpatrick and Klawe [163] carry out the details of the F_m, G_m problem for c=1 for alphabetic codes in the nonbinary case thus addressing in part an issue raised in [134]. They also provide a generalized entropy-type upper bound for this F,G pair.

Occasionally the Huffman problem is considered under some other optimization criterion variant besides those in Parker and Knuth for which a generalized merge algorithm finds the optimal code tree. One such problem is to find Cover's [164] competitively optimal code tree. The Huffman tree finds L such that $\Sigma p(i)(l(i)-l'(i))<0$ for all L'. Cover asks for the tree such that $\Sigma p(i)sgn(l(i)-l'(i))<0$. Feder [165] and Yamamoto and Itoh [166] also examine competitive optimality, and Yamamoto and Itoh show that if the competitively optimal code exists for given source probabilities then it is also the Huffman code tree, and they provide conditions on P for its existence. Perhaps it may be of interest to consider the tree such that $\Sigma p(i)f(l(i)-l'(i))<0$ for some general class of nonlinear functions f.

Larmore [167] is interested in minimizing a particular nonlinear function of average codeword length and codeword variance which arises in modeling delay in communicating coded strings. He resolves the Huffman version of his problem but leaves the Hu-Tucker version open. One can imagine a broader class of nonlinear functions of average codeword length and other codeword length functionals being of interest as performance criteria as well. It would be of interest to determine how broadly Larmore's method applies. Since the approach consists of two stages, one which minimizes all linear combinations of average codeword length and codeword variance, and one which searches over all candidate trees within a constrained set determined by the result of the first stage together with the form of the particular nonlinear function in the problem formulation, it seems reasonable to expect that the general approach is more widely applicable. Nevertheless the algorithm itself is not as elegant as the merge-based Huffman algorithm.

Cohen and Fredman [168] address the minimization of a performance criterion which arises when m independent processors use a single test tree in parallel. Each is to identify its own distinguished item from among the K items. In particular, m items are selected independently according to P. Assuming each processor's search must be completed before the next phase of the computation begins, the cost to be minimized is the average length of the maximum of the m path lengths through the tree. For m=1, this is the ordinary Huffman problem or the ordinary Hu-Tucker problem if an alphabetic constraint is imposed. The approach of their suboptimal algorithm is to derive a set of transformed probabilities to which the m=1 Huffman algorithm is applied. Clearly it would be of interest to resolve this intriguing problem

optimally. Similarly one can introduce other search problems based on Cohen and Fredman's scenario of m independent processors, and these may also be of interest.

II.7. Coding with Unequal Cost Code Symbols: The Karp Problem

In the binary unequal costs coding problem, using the code symbol 0 costs $c(0) \geq 1$ and using the code symbol 1 costs $c(1) \geq 1$. Thus a path through the code tree which consists of M(i) left branches and N(i) right branches, or, that is, a codeword consisting of M(i) 0's and N(i) 1's, costs $l(i)=c(0)M(i)+c(1)N(i)$. When $c(0)=c(1)=1$, l(i) is codeword length in the usual sense. Karp [169] first gave an algorithm to find the optimal code tree in the sense of minimum average codeword cost, that is to solve $\min_L \Sigma_{1 \leq i \leq \kappa} p(i)l(i)$, and this problem will be referred to as the Karp problem. Actually, Karp addressed the case of r-ary code alphabets and distinguished between exhaustive codes, in which each interior node of the tree has exactly r descendants, and nonexhaustive codes, in which each interior node has at least 2 and no more than r descendants. Generally in unequal costs coding problems, the algorithms to find optimal exhaustive codes are simpler but the nonexhaustive codes yield lower average cost. For binary code alphabets the distinction does not arise. For $c(0)=c(1)=1$, Karp's problem is Huffman coding.

Karp's method is an integer programming method which makes use of a useful structure function method for representing code trees algebraically. Golin and Rote [170] provide a dynamic programming algorithm to solve the Karp problem which is much more computationally efficient than Karp's approach. Both Karp and Golin and Rote address the unordered or Huffman version of the unequal costs coding problem. See Itai [146] for the ordered or Hu-Tucker version.

Entropy bounds for the Karp problem are given by Krause [171]. See also [172]. The Kraft inequality for unequal costs on which the bounds are based is found in [169]. See also [173]. Recently De Prisco and Persiano [174] extended the Kraft-inequality-like results of [123] for the Hu-Tucker problem to the ordered Karp problem.

A number of authors have dealt with techniques for finding good but not optimal unequal costs codes for arbitrary source probabilities [175-177] or have developed techniques to find optimal codes for uniform source probabilities and special cost structures [178-181].

Optimal codes for uniform source probabilities and arbitrary cost structures is a problem of interest for which a substantial body of results are known. Varn [182] first provided algorithms to find the optimal exhaustive and nonexhaustive code trees in the minimum average codeword cost sense. Perl, Garey, and Even [183] introduced a nonexhaustive algorithm which is computationally better than Varn's and provided a simpler algorithm to find the optimal nonexhaustive code tree in the minimax codeword cost sense. In the exhaustive case the same code trees are optimal according to both minimum average and minimax cost performance criteria [184]. Other nonexhaustive algorithms are in [185-186]. Choi and Golin's [187] recent algorithm for the nonexhaustive Varn problem is the best computationally. Savari [188] provides performance bounds on the resulting average codeword cost of Varn codes. See also [189] for the binary case.

Varn's algorithm for exhaustive code trees is a top down splitting algorithm in which a least cost node is replaced by two descendant nodes at each stage. The costs of the descendants are the cost of the parent plus the cost of the associated branch. The sequence of Varn code trees has an underlying Fibonacci tree structure first identified by Horibe [190] in the case that $c(0)=1$ and $c(1)=2$ and since generalized for arbitrary integer costs [191-192]. A similar generalized Fibonacci structure is present in the sequence of nonexhaustive minimax Varn trees [193]. Hinderer [194] also addresses the binary Varn problem independently.

Horibe has written a number of papers on the properties, particularly the balance properties, of the Fibonacci trees arising in the solution to the Varn problem [195-198]. One property of particular interest is the fact that the generalized Fibonacci tree which solves the Varn problem for arbitrary costs both minimizes average codeword cost and maximizes the entropy of the probability distribution induced by the tree [199]. The induced distribution is of the form $P=\{p(i)=t^{l(i)}, i=1,\ldots,K\}$ where $t^{c(0)}+t^{c(1)}=1$. The idea of induced distribution will reappear in the following in the discussion of parse trees.

Besides uniform sources, Karp's problem is also easily solved for distributions whose relationship to the code symbol costs generalizes the relationship between dyadic sources and binary equal code symbol costs [200]. These distributions can be coded with zero redundancy.

Consider now the generalized unequal costs coding problem in which the costs of the code symbols vary with the position of the code symbol in the codeword. This model of code symbol costs is Shannon's original discrete noiseless channel model in which a finite state diagram describes the cost of each code symbol when it appears following some particular string of code symbols in the codeword [201]. Finite state channel models arise frequently in the literature on magnetic and optical recording [202], however this literature is not typically concerned with either variable-length-to-block codes or with minimum average cost as a performance criterion. Csiszár [203] proposed a nonoptimal approach to variable length coding for finite state code symbol costs and provided entropy-type performance bounds. The particular case of costs which grow rapidly with the position of the code symbol in the codeword was addressed in [204-205], and bounds of Csiszár's form are shown not to hold. It would be of interest to derive algorithms for minimum average codeword cost codes for finite state cost models, for both arbitrary source probabilities and for the special case of uniform source probabilities. Recently the author [206] was able to show that a Varn-like algorithm applies to certain finite state cost models for uniform source probabilities. In essence $c(0)+c(1)$ must be nearly constant at every interior node of the code tree for the Varn-like algorithm to apply. Recall that for Varn codes, $c(0)$ and $c(1)$ are fixed at every interior node of the code tree.

II.8. Coding for Bidirectionality and Synchronization

In some coding applications it is of interest to select the assignment of patterns of 0's and 1's within the codewords in order to obtain good performance with respect to some secondary pattern-based criterion. Ideally the minimum average codeword length property of the underlying code tree is not affected by the choice among codewords with the specified codeword lengths but sometimes average codeword length is sacrificed in order to gain improved performance with respect to the secondary criterion.

Bidirectionality is one such secondary criterion. Several authors have considered this problem independently [207-210]. If the set of codewords also has the property that when read backwards the codewords form a code tree, then the code is bidirectional. Questions of interest include characterizing the sets of codeword lengths which are consistent with bidirectionality, finding codeword string assignments with the bidirectionality property when they exist, bounding the increase in average codeword length required to ensure bidirectionality when the Huffman codeword lengths are inconsistent with bidirectionality. Synchronization is another such secondary criterion.

A number of authors [211-217] have examined conditions on the set of codeword lengths sufficient for the existence of one or more synchronizing codewords, a codeword whose pattern of code symbols, relative to the other codewords in the code, is such that the next received symbol is always correctly interpreted as the beginning of a new codeword, whether or not the synchronizing codeword was itself correctly received and/or decoded. Other authors [218-219] require the synchronizing codeword to have the stronger property that it is itself always correctly decoded when correctly received. In each case there is interest in constructing the codeword set and in evaluating both its synchronizing performance and its

loss of average codeword length relative to the Huffman code if that loss occurs.

An alternative perspective on synchronization is to design a set of variable length codes for a given source alphabet size with good overall synchronization properties and then to select the best code from within that set, best in the sense of minimum average codeword length for the source distribution. This is the approach taken by Titchener [220-222] and Higgie [223-225] and further examined by Swaszek and Willett [226] under the name of "T-codes". Unfortunately a good analytical evaluation of the tradeoff between synchronization performance and average codeword length for T-codes is not available, nor is there an algorithm other than exhaustive search for the identification of the best T-code in some combined synchronization/average-codeword-length sense.

Several authors have addressed the issue of synchronization recovery in variable-length codes beginning with Maxted and Robinson [227]. See Swaszek and DiCicco [228] and the references therein.

Another application in which the patterns of code symbols within the codewords matter is in using Huffman codes cryptographically [229]. One of the issues here is to identify conditions on the Huffman codeword lengths such that different strings of source symbols lead to an identical string of code symbols when the assignment of code symbols within the codewords differs. This kind of ambiguity is of cryptographic significance.

III. Parse Trees

In variable-length-to-fixed binary source coding, a parse tree is used to associate a string of binary source symbols with each of a set of K code symbols. A parse tree is a complete binary tree with K leaf nodes and K-1 internal nodes including the root at the top of the tree. Pairs of branches descend from each internal node. Left branches are labeled with the source symbol 0 and right branches are labeled with the source symbol 1. Each leaf node is labeled with one of the K code symbols. A path through the parse tree from the root to a leaf describes the string of source symbols associated with the code symbol at the leaf. The problem is to find a parse tree to optimize some performance criterion, possibly under the imposition of constraints on the form of the tree, when the source symbol is 0 with probability r_0 and 1 with probability r_1. Thus a path through the parse tree which consists of M(i) left branches and N(i) right branches, or, that is, a parse string consisting of M(i) 0's and N(i) 1's, corresponds to a string of source symbols which occurs with probability $R=\{r(i)=r_0^{M(i)}\ r_1^{N(i)},\ i=1,\ldots,K\}$. Refer to such an R as the probability distribution induced by the parse tree.

III.1.Tunstall Parse Trees and Variants

Tunstall parsing [6, 230], also derived independently by Verhoeff [231], solves the problem of finding the parse tree to maximize average parse string length, or, that is, to solve $\max_{M,N}\Sigma_{1\le i\le K} r(i)(M(i)+N(i))$ for $M=\{M(i),\ i=1,\ldots,K\}$, $N=\{N(i),\ i=1,\ldots,K\}$. Here $l(i)=M(i)+N(i)$ is the length of the i^{th} parse string, and it arises with probability r(i). There are no further constraints on the problem however $\Sigma 2^{-l(i)}\le 1$ must hold in order that the binary parse tree exists, from the Kraft inequality. Another way of thinking about Tunstall parsing is that it finds the parse tree whose induced distribution R has maximum entropy, or, that is, it solves $\max_R -\Sigma r(i)\log_2 r(i)$. This follows from the fact that [230] $H_I(R) = EL\ H_D(r)$, where $H_I(R)=-\Sigma r(i)\log_2 r(i)$ is the entropy of the distribution induced by a parse tree, $H_D(r)=-r_0\log_2 r_0-r_1\log_2 r_1$ is the entropy of the source distribution, and EL is the expected parse string length of that parse tree.

A minimum discrimination perspective on Tunstall parsing is due to Stubley and Blake [232] who examined a broader class of parsing problems. In their framework, the distribution induced by the parse tree, R, is the closest induced distribution to the uniform distribution $S=\{s(i)=1/K,\ i=1,\ldots,K\}$ in the

minimum discrimination sense. That is to say, Tunstall parsing solves $\min_{M,N}\Sigma r(i)\log_2 r(i)/s(i)$ for S uniform. Since S is a constant with respect to the minimization, this is exactly maximum entropy parsing. More generally, Stubley and Blake consider the minimum discrimination parsing problem in which S is a given distribution, not necessarily uniform, to be imposed on the leaves of the parse tree, and the goal is for the probability distribution induced by the parse tree to be as close as possible to a desired code symbol distribution S. They do not have an algorithm to find the optimal parse tree for the problem in its full generality.

Tunstall's algorithm is a top down splitting algorithm in which a node of greatest probability is replaced by two descendant nodes at each stage. The weights of the descendants are the weight of the parent multiplied by the probability of the associated branch. Tunstall's algorithm is exactly isomorphic to Varn's algorithm for exhaustive unequal costs code trees for the uniform source distribution under the mapping $r_0=t^{c(0)}$, $r_1=t^{c(1)}$ as pointed out by Savari [188]. Both algorithms maximize the entropy of the induced distribution.

Petry and Schalkwijk [233-234] identify a generalized Fibonacci structure in binary Tunstall parse trees and exploit it in the implementation process. Actually their parsing problem was formulated independently of Tunstall but can be identified as equivalent.

A recent paper by Fabris, Sgarro and Pauletti [235] addresses some characterization results for Tunstall codes, such as the relationship between source probabilities and the resulting minimum and maximum parse string lengths, motivated by applications to adaptive Tunstall parsing.

Nonexhaustive parse trees are not usually of interest, because every possible string of source symbols must be encoded, however Algra [236] describes a particular nonoptimal nonexhaustive parse tree variant of Tunstall parsing which incorporates special escape strings pointing to codewords for selected unlikely source strings. Strictly speaking Algra's algorithm is not a variable-length-to-fixed algorithm but rather a special type of variable-to-variable-length coding algorithm in which the variable length parse strings are represented by either single code symbols or pairs of code symbols in a particular highly structured way well suited to implementation. However, in general, the compression performance can be improved by using dual tree codes of less restricted form as described in a later section.

Lempel, Even, and Cohn [237] examine a parsing problem for the special case that the binary source symbols are equally probable and thus the induced distribution R is dyadic. One can imagine a generalization of Lempel, Even, and Cohn's problem in which r_0 and r_1 are not necessarily equal to ½, and Tunstall parsing would be a special case of this generalization. The particular parsing problem of Lempel et al. is to solve $\min_{M,N}\Sigma r(i)\log_2 s(i)/\Sigma r(i)\log_2 r(i)$ where R is the dyadic distribution induced by the parse tree and S is a given distribution imposed on the leaves of the tree. This is in a sense a minimum distance parsing problem for a notion of distance different from discrimination. Lempel et al. provide an algorithm to find the optimal parse tree for their problem which is an iterative algorithm which converges to the optimal tree, and, in each stage of the algorithm, a Huffman-type merge is performed. In their original problem statement, Lempel et al. associate costs with the individual code symbols and minimize the ratio of average code symbol cost to parse string length.

Jelinek and Schneider [238] address a parse problem in which the performance criterion is to minimize the average value of a constant c raised to the parse string length. But, because the average is taken with respect to the distribution induced by the parse tree in which $r(i)=r_0^{M(i)}r_1^{N(i)}$, it is immediate that the Tunstall splitting algorithm applies to this problem as well, just by replacing the source probabilities r_0 and r_1 with cr_0 and cr_1 respectively.

In the case that Tunstall's splitting algorithm is applied to "probabilities" r_0 and r_1 where $r_0+r_1<1$, the resulting tree also minimizes $\Sigma r(i)$, a quantity identically equal to 1 for all parse trees when $r_0+r_1=1$ [239].

Parse problems have been much less extensively studied than code problems. It may be that interesting parse problem variants are amenable to analysis.

III.2. Dualities Between Parse Trees and Unequal Costs Code Trees

The isomorphism between binary Tunstall parse trees and binary Varn code trees is a special case of a more general duality between parse trees and unequal costs code trees [240]. In particular, the general parse problem of Stubley and Blake can be formulated as finding the tree to solve $\min_R D(R,S)=\min_R \Sigma r(i)\log_2 r(i)/s(i)$ where R is the distribution induced by the tree and S is a given distribution imposed on the leaves of the tree. With this perspective on parse problems in mind, we can reformulate the minimum average codeword cost problem of Karp as finding the tree to solve $\min_Q D(P,Q)$ where Q is the distribution induced by the code tree under the mapping $q(0)=t^{c(0)}$, $q(1)=t^{c(1)}$ and P is a given distribution imposed on the leaves of the tree, the source distribution. Specifically $Q=\{q_i=(t^{c(0)})^{M(i)}(t^{c(1)})^{N(i)}=t^{l(i)},\ i=1,...,K\}$. Note that $\min_Q(P,Q)$ is equivalent to $\min_{M,N}\Sigma p(i)(M(i)+N(i))$ because the entropy of P is a constant with respect to the minimization. The same algorithm does not solve both the Stubley and Blake problem and the Karp problem due to the asymmetry of D with respect to its arguments.

A similar duality can he established between Karp coding and generalized Lempel, Even, Cohn parsing [237] in which Huffman coding and the original, dyadic, parsing problem of Lempel et al. are dual to each other.

III.3. Dual Tree Coding

In variable-to-variable-length binary coding, or dual tree coding, a parse tree with K leaf nodes is matched to a code tree with K leaf nodes to optimize some performance criterion. Ideally it would be of interest to minimize the ratio of average codeword length, or more generally average codeword cost, to average parse string length, however, that problem has not been tractable, and a number of authors have examined the alternative performance criterion of $\min_{R,Q} D(R,Q)$ where R is the distribution induced by the parse tree and Q is the distribution induced by the code tree [241-243]. In both of these equal costs dual tree coding problems the code tree will be the Huffman tree for the distribution induced by the parse tree so the problem reduces to finding the optimal parse tree. In general, minimum discrimination dual tree coding is not equivalent to minimizing the ratio of average codeword length to average parse string length, but for dyadic R they are the same dual tree codes. Unfortunately minimum discrimination dual tree coding is also intractable for arbitrary source distributions, although good approximate solutions are known.

III.4. Parse Trees and Random Number Generation

Given a parse tree and an associated source distribution, a distribution is induced at the leaves of the tree. Label each leaf with one of $N \leq K$ variables. Associate the sum of the induced probabilities at the leaves which have a common label with that variable. Thus a random variable whose distribution is the source distribution can be used to generate random variables according to the labeled leaf distribution by parsing a string of source symbols according to the labeled parse tree. A problem originated by Knuth and Yao [244] and discussed in [24] is to start with the distribution of the random variable to be generated and the source distribution and construct the parse tree which minimizes the expected number of source symbols used to generate the random variable. An optimal parse tree for an equiprobable binary source can be found from the coefficients of the expansions base ½ of the desired probabilities. The generalization to arbitrary discrete sources appears in [245]. Of course there is also a substantial literature on random number generation based on a variety of techniques other than parse trees.

IV. Representing and Counting Code and Parse Trees

It may be desirable to represent the code or parse tree itself as a string of symbols in an unambiguously recoverable fashion. Of course the efficiency of such a representation is of interest, both in the sense of the compactness of the representation as well as in the sense of the computational effort required to recover the tree from the representation. One can always represent the tree by the probabilities from which it was generated, assuming that the algorithm from which the tree was generated, together with tie-breaking conventions, is known. Even if the tree is not a Huffman tree, it can be represented in this way by the dyadic probabilities corresponding to the tree path lengths. However, because this tends to be a computationally intensive approach, one of the other, general, tree representation techniques may be useful for code and parse tree representation. These techniques do not exploit any distributional information on the set of all potential code trees, nor is such distributional information typically available. Some of the tree representation literature is focused on structures which enable fast encoding or, more typically, decoding. There is a larger literature on implementation issues, particularly for Huffman codes, which is not included here.

Surveys of tree representation techniques are given by Katajainen and Makinen [246-247]. They categorize the various techniques as permutation-based, rotation-based, and enumeration-based. Permutation-based methods involve a traversal of the tree identifying interior and leaf nodes distinctively. Rotation-based methods describe a series of transformations to take a canonical tree into the given tree. Enumeration-based methods describe a technique for generating all trees in a fixed order, and denote the given tree by its index in the list of trees so generated. Labels can be separated from tree shapes and encoded separately.

The structure function idea described by Karp [169] resembles the permutation-based approaches outlined in [246-247] and can be used as a tree representation method.

The canonical code tree of Schwartz and Kallick [248] is essentially a rotation-based method which has appeared in various forms, sometimes independently, in the code tree literature [249-253]. In an interesting recent paper, Parker and Ram [254] construct a lattice of binary trees based on a partial order which describes relative balance. Edges in the lattice correspond to the exchange of a pair of sibling leaves at one point in the tree for a pair elsewhere, a rotation-type operation. The Huffman algorithm can be recovered as the solution to an optimization problem in this lattice-theoretic framework. It may also be that lattice concepts will provide new insights into other optimal coding problems.

A series of papers on counting the number of distinct code trees [55, 140, 255-262] yields expressions useful in the evaluation of the efficiency of tree representation techniques relative to enumeration based techniques. This is a subset of a much larger literature on counting the number of discrete structures of various types.

Bobrow and Hakimi [263] and Fitingof and Waksman [264] have described methods for representing code and parse trees as graphs by identifying common subtrees with each other. Graph representations may not be as useful as string representations in transmitting or storing code trees. Trees with common subtree structures may lead to string representations which can themselves be further compressed.

One can also consider the possibility of representing a tree, as a special case of a graph or digraph, by means of its adjacency matrix, and then treating that matrix of 0's and 1's as a bit map for compression purposes, e.g. [18, Sect. 3.5].

V. Concluding Remarks

There are many interesting variants of coding and parsing problems for which further investigation

may be worthwhile, both from an algorithmic perspective as well as from the perspective of performance evaluation and tree characterization. It is hoped that this survey is a useful guide to the existing literature.

References

1. D.A. Huffman, "A method for the construction of minimum-redundancy codes," Proc. of the IRE, Vol. 40, pp. 1098-1101, Sep. 1952.
2. R. Ahlswede and I. Wegener, Search Problems, Wiley, New York, 1987.
3. M. Aigner, Combinatorial Search, B.G. Teubner, Stuttgart;Wiley, New York, 1988.
4. K. Hinderer and M. Stieglitz, "On polychotomous search problems," European J. of Operations Research, Vol. 73, pp. 279-294, 1994.
5. R. Hassin and M. Henig, "Monotonicity and efficient computation of optimal dichotomous search," Discrete Appl. Math., Vol. 46, pp. 221-234, 1993.
6. B.P. Tunstall, "Synthesis of noiseless compression codes," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, GA, 1968
7. J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," IEEE Trans. on Inform. Th., Vol. IT-23, No. 3, pp. 337-343, May 1977.
8. J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," IEEE Trans. on Inform. Th., Vol. IT-24, No. 5 pp. 530-536, Sep. 1978.
9. R.G. Gallager, "Variations on a theme by Huffman," IEEE Trans. on Inform.Th., Vol. IT-24, No. 6, pp. 668-674, Nov. 1978.
10. G.V. Cormack and R.N. Horspool, "Algorithms for adaptive Huffman codes," Inform. Processing Letters, Vol. 18, pp. 159-165, 1984.
11. D.E. Knuth, "Dynamic Huffman coding," J. of Algorithms, Vol. 6, pp. 163-180, 1985.
12. J.S. Vitter, "Design and analysis of dynamic Huffman codes," JACM, Vol. 34, No. 2, pp. 825-845, Oct. 1987.
13. D.A. Lelewer and D.S. Hirschberg, "Data compression," ACM Computing Surveys, Vol.19, No. 3, pp. 261-296, Sep. 1987.
14. R.M. Gray, M. Cohn, L.W. Craver, A. Gersho, T. Lookabaugh, F. Pollara, M. Vetterli, Non-US Data Compression and Coding Research, Foreign Applied Sciences Assessment Center Technical Assessment Report, SAIC, McLean, VA, Nov. 1993.
15. J.A. Storer, ed., Image and Text Compression, Kluwer, Boston, 1992.
16. K. Sayood, Introduction to Data Compression, Morgan Kaufmann, San Francisco, CA, 1996.
17. T.J. Lynch, Data Compression Techniques and Applications, Van Nostrand Reinhold, NY, 1985.
18. I.H. Witten, A. Moffat, and T.C. Bell, Managing Gigabytes, Van Nostrand Reinhold, NY, 1994.
19. T.C. Bell, J.G. Cleary, and I.H. Witten, Text Compression, Prentice Hall, Englewood Cliffs, NJ, 1990.
20. T.C. Bell, I.H. Witten, and J.G. Cleary, "Modeling for text compression," Computing Surveys, Vol. 21, No. 4, pp. 557-592, Dec. 1989.
21. J.A. Storer, Data Compression: Methods and Theory, Computer Science Press, 1988.
22. R.G. Gallager, Information Theory and Reliable Communication, Wiley, New York, 1968.
23. R.J. McEliece, The Theory of Information and Coding, Addison-Wesley, Reading, MA, 1977, second printing with revisions, 1979.
24. T.M. Cover and J.A. Thomas, Elements of Information Theory, Wiley, New York, 1991.

25. I. Csiszár and J. Körner, Information Theory: Coding Theorems for Discrete Memoryless Systems, Academic, New York, 1981.
26. E. Horowitz and S. Sahni, Fundamentals of Computer Algorithms, Computer Science Press, 1978.
27. A.V. Aho, J.E. Hopcroft, and J.D. Ullman, Data Structures and Algorithms, Addison-Wesley, Reading, MA, 1983, reprinted with corrections 1987.
28. R. Sedgewick, Algorithms, Addison-Wesley, Reading, MA, 1983.
29. D.E. Knuth, The Art of Computer Programming, Vol. 3, Sorting and Searching, Addison Wesley, Reading, MA, 1973.
30. T.C. Hu, Combinatorial Algorithms, Addison Wesley, Reading, MA, 1982.
31. K.A. Ross and C.R.B. Wright, Discrete Mathematics, Prentice-Hall, Englewood Cliffs, NJ, 1985.
32. R.P. Grimaldi, Discrete and Combinatorial Mathematics, 2nd Ed., Addison-Wesley, Reading, MA, 1989.
33. G.O.H. Katona and T.O.H. Nemetz, "Huffman codes and self-information," IEEE Trans. on Inform. Th., Vol. IT-22, No. 3, pp. 337-340, May 1976.
34. G.O.H. Katona and M.A. Lee, "Some remarks on the construction of optimal codes," Acta Math. Academiae Scientiarum Hungaricae, Vol. 23, No. 3-4, pp. 439-442, 1972.
35. T. Nemetz, "On the word-lengths of Huffman codes, Problems of Control and Inform. Theory, Vol. 9, No. 4, pp. 231-242, 1980.
36. G. Longo and T. Nemetz, "Once more on the word-length of optimal source codes," Trans. of the Ninth Prague Conf. on Information Theory, Statistical Decision Functions, Random Processes, Prague, 28 Jun. - 2 Jul. 1982, Vol. B. pp. 63-69, D. Reidel, Boston, MA, 1983.
37. T. Nemetz and J. Simon, "Self-information and optimal codes," Colloquia Mathematica Societatis János Bolyai, 16, Topics in Information Theory, Keszthely, Hungary, 1975, I. Csiszár and P. Elias, eds., pp. 457-468, North Holland, New York, 1977.
38. M. Buro, "On the maximum length of Huffman codes," Inform. Processing Letters, Vol. 45, pp. 219-223, 2 Apr. 1993.
39. K. Birmiwal, "The minimum codeword length and redundancy in the binary Huffman code for uncertain sources," IEEE Trans. on Inform. Th., Vol. 36, No.2, pp. 439-443, Mar. 1990.
40. R.M. Capocelli and A. De Santis, "A note on D-ary Huffman codes," IEEE Trans. on Inform. Th., Vol. 37, No. 1, pp. 174-179, Jan. 1991.
41. K.C. Chu and J. Gill, "Upper bounds on Huffman codeword lengths," Proc. of the IEEE Intl. Symp. on Inform. Th., Budapest, Hungary, p. 111, 24-28 Jun. 1991.
42. R. Schack, "The length of a typical Huffman codeword," IEEE Trans. on Inform.Th., Vol. IT-40, No. 4, pp. 1246-1247, Jul. 1994.
43. J.F. Cheng, S. Dolinar, M. Effros, and R. McEliece, "Data expansion with Huffman codes," Proc. IEEE Intl. Symp. on Inform. Th., Whistler, BC, p. 325, 1995.
44. R. De Prisco and A. De Santis, "A new bound for the data expansion of Huffman codes," IEEE Trans. on Inform. Th., to appear, Nov. 1997.
45. B.L. Montgomery and B. V. K. Vijaya Kumar, "On the average codeword length of optimal binary codes for extended sources," IEEE Trans. on Inform. Th., Vol. IT-33, No. 2, pp. 293-296, Mar. 1987.
46. D. Manstetten, "Tight bounds on the redundancy of Huffman codes," IEEE Trans. on Inform. Th., Vol. 38, No. 1, pp. 144-151, Jan. 1992.
47. R.M. Capocelli and A. De Santis, "Tight upper bounds on the redundancy of Huffman codes," IEEE Trans. on Inform. Th., Vol. 35, No. 5, pp. 1084-1091, Sep. 1989.

48. R.M. Capocelli and A. De Santis, "Variations on a theme by Gallager," Image and Text Compression, J.A. Storer, ed., pp. 181-213, Kluwer, Boston, MA, 1992.
49. R.M. Capocelli and A. De Santis, "New bounds on the redundancy of Huffman codes," IEEE Trans. on Inform. Th., Vol. 37, No. 4, pp. 1095-1104, Jul. 1991.
50. O. Johnsen, "On the redundancy of binary Huffman codes," IEEE Trans. on Inform. Th., Vol. IT-26, No. 2, pp. 220-222, Mar. 1980.
51. R.M. Capocelli, R. Giancarlo, and I.J. Taneja, "Bounds on the redundancy of Huffman codes," IEEE Trans. on Inform. Th., Vol. IT-32, No. 6, pp. 854-857, Nov. 1986.
52. B.L. Montgomery and J. Abrahams, "On the redundancy of optimal binary prefix-condition codes for finite and infinite sources," IEEE Trans. on Inform.Th., Vol. IT-33, No. 1, pp. 156-160, Jan. 1987
53. R. De Prisco and A. De Santis, "On the redundancy achieved by Huffman codes," Inform. Sciences, Vol. 88, pp. 131-148, 1996.
54. R.W. Yeung, "Local redundancy and progressive bounds on the redundancy of a Huffman code," IEEE Trans. on Inform. Th., Vol. 37, No. 3, pp. 687-691, May 1991.
55. S.W. Golomb, "Sources which maximize the choice of a Huffman coding tree," Inform. and Control, Vol. 45, pp. 263-272, 1980.
56. N.C. Geçkinli, "Two corollaries to the Huffman coding procedures," IEEE Trans. on Inform. Th., Vol. IT-21, No. 3, pp. 342-344, May 1975.
57. A.B. Vinokur, "Huffman codes and maximizing properties of Fibonacci numbers," Cybernet. Systems Anal., Vol. 28, No. 3, pp. 329-334, 1993, translated from Kibernetikai Systemnyi Analiz, Vol. 187, No. 3, pp. 10-15, May-Jun. 1992.
58. Y. Horibe, "Balance properties of optimal binary trees," Proc. of the IEEE Intl. Symp. on Inform. Th., Budapest, Hungary, p. 110, 24-28 Jun. 1991.
59. D.S. Hirschberg, L.L. Larmore, and M. Molodowitch, "Subtree weight ratios for optimal binary search trees," UC Irvine Technical Report No. 86-02, 29 Jan. 1986.
60. P. Kirrinnis, "An optimal bound for path weights in Huffman trees," Inform. Processing Letters, Vol. 51, pp. 107-110, 1994.
61. M. Jakobsson, "Huffman coding in bit-vector compression," Inform. Processing Letters, Vol. 7, No. 6, pp. 304-307, Oct. 1978.
62. P.R. Stubley, "On the redundancy of optimum fixed-to-variable length codes," Proc. Data Compression Conf., Snowbird, UT, 29-31 Mar. 1994, J.A. Storer and M. Cohn, eds., pp. 90-97, IEEE Computer Society Press, Los Alamitos, CA, 1994.
63. P.R. Stubley, "The redundancy of optimum codes of multinomially distributed sources," preprint, 14 Oct. 1994.
64. P.M. Fenwick, "Huffman code efficiencies for extensions of sources," IEEE Trans. on Communications, Vol. 43, No. 2/3/4, pp. 163-165, Feb./Mar./Apr. 1995.
65. T. Berger and X. Zhang, "On Huffman codes for extension alphabets," Proc. IEEE Inform. Th. Workshop, Israel, 9-13 Jun. 1996.
66. R.E. Kricevskii, "The block length necessary to obtain a given redundancy," Soviet Math. Dokl., Vol. 7, No. 6, pp. 1416-1420, 1966, translated from Dokl. Akad. Nauk SSSR, Vol. 171, No. 1, 1966.
67. F.K. Hwang, "On finding a single defective in binomial group testing," JASA, Vol. 69, No. 345, pp. 146-150, 1974.
68. Y.C. Yao and F.K. Hwang, "On optimal nested group testing algorithms," J. Statistical Planning and Inference, Vol. 24, pp. 167-175, 1990.

69. R. Hassin, "A dichotomous search for a geometric random variable," Operations Research, Vol. 32, No. 2, pp. 423-439, 1984.
70. C.G. Günther and W.R. Schneider, "Entropy as a function of alphabet size," Proc. IEEE Intl. Symp. on Inform. Th., San Antonio, TX, p. 70, Jan. 1993.
71. L.L. Campbell, "Averaging entropy," IEEE Trans. on Inform. Th., Vol. 41, No. 1, pp. 338-339, Jan. 1995.
72. B.L. Montgomery, H. Diamond, and B.V.K. Vijaya Kumar, "Bit probabilities of optimal binary source codes," IEEE Trans. on Inform. Th., Vol. 36, No. 6, pp. 1446-1450, Nov. 1990.
73. M. Gutman, "Fixed-prefix encoding of the integers can be Huffman-optimal," IEEETrans. on Inform. Th., Vol. 36, No. 4, pp. 936-938, Jul. 1990.
74. A.C. Tucker, "The cost of a class of optimal binary trees," J. Combinatorial Theory (A), Vol. 16, pp. 259-263, 1974.
75. F.K. Hwang, "An explicit expression for the cost of a class of Huffman trees," Discrete Mathematics, Vol. 32, pp. 163-165, 1980.
76. H.K. Hwang, "Optimal algorithms for inserting a random element into a random heap," IEEE Trans. on Inform. Th., Vol. 43, No. 2, pp. 784-787, Mar. 1997.
77. G. Longo and G. Galasso, "An application of informational divergence to Huffman codes," IEEE Trans. on Inform. Th., Vol. IT-28, No. 1, pp. 36-43, Jan. 1982.
78. F. Fabris, "The Longo-Galasso criterion for Huffman codes: an extended version," AltaFrequenza, Vol. LVIII, No. 1, pp. 35-38, Jan. - Feb. 1989.
79. A.C. Blumer, "Minimax universal noiseless coding for unifilar and Markov sources," IEEE Trans. on Inform. Th., Vol. IT-33, No. 6, pp. 925-930, Nov. 1987.
80. M.B. Pursley and L.D. Davisson, "Mismatch bounds for variable rate source codes with applications to universal data compression," Proc. AFOSR Workshop in Communications Theory and Applications, Provincetown, MA, pp. 33-37, Sep. 1978.
81. T. Nemetz, "Information type measures and the applications to finite decision problems," Carlton Mathematical Lecture Notes, No. 17, Carleton University, Ottawa, May 1977.
82. T.C. Hu and A.C. Tucker, "Optimum binary search trees," Proc. Second Chapel Hill Conf. on Combinatorial Mathematics and Its Applications, Chapel Hill, NC, pp. 285-305, May 1970.
83. F.K. Hwang, "Generalized Huffman trees," SIAM J. Appl. Math., Vol. 37, No. 1, pp. 124-127, Aug. 1979.
84. K.C. Chu and J. Gill, "Mixed-radix Huffman codes," Proc. 11th Intl.Conf. of the Chilean Computer Science Society, Santiago, Chile, 15-18 Oct. 1991, R. Baeza-Yates and U. Manber, eds., pp. 209-218, Plenum, New York, 1992.
85. R.G. Gallager and D.C. Van Voorhis, "Optimal source codes for geometrically distributed alphabets," IEEE Trans. on Inform. Th., Vol. IT-21, No. 2, pp. 228-230, 1975.
86. S.W. Golomb, "Run-length encodings," IEEE Trans. on Inform. Th., Vol. IT-12, pp. 399-401, Jul. 1966.
87. P. Humblet, "Optimal source coding for a class of integer alphabets," IEEE Trans. on Inform. Th., Vol. IT-24, No. 1, pp. 110-112, 1978.
88. J. Abrahams, "Huffman-type codes for infinite source distributions," J. Franklin Institute, Vol. 331B, No. 3, pp. 265-271, 1994.
89. A. Kato, T.S. Han, and H. Nagaoka, "Huffman coding with an infinite alphabet," IEEETrans. on Inform. Th., Vol. 42, No. 3, pp. 977-984, May 1996.

90. N. Merhav, G. Seroussi, and M.J. Weinberger, "Optimal prefix codes for the two-sided geometric distributions," Proc. IEEE Intl. Symp. on Inform. Th., Ulm, Germany, p. 71, 29 Jun.-4 Jul. 1997.

91. T. Linder, V. Tarokh, and K. Zeger, "Existence of optimal prefix codes for infinite source alphabets," IEEE Trans. on Inform. Th., submitted.

92. A. Kato, "Huffman-like optimal prefix codes and search codes for infinite alphabets," IEEE Trans. on Inform. Th., submitted.

93. S.J. Golin, "A simple variable-length code," Signal Processing, Vol. 45, pp. 23-35, 1995.

94. P.S. Yeh, R. Rice, and W. Miller, "On the optimality of code options for a universal noiseless coder," NASA Jet Propulsion Laboratory Publication 91-2, 15 Feb 1991.

95. J. Teuhola, "A compression method for clustered bit-vectors," Inform. ProcessingLetters, Vol. 7, No. 6, pp. 308-311, Oct. 1978.

96. K.M. Cheung and P. Smyth, "A high-speed distortionless predictive image compression scheme," Proc. 1990 Intl. Symp. on Inform. Th. and Its Applications, Honolulu, pp. 467-470, 27-30 Nov. 1990, and K.-M. Cheung, P. Smyth, and H. Wang, "A high-speed distortionless predictive image compression scheme," NASA Jet Propulsion Laboratory TDA Progress Report 42-102, 15 Aug. 1990.

97. K.M. Cheung and P. Smyth, "An efficient source coding scheme for progressive image transmission," Proc. IEEE Intl. Symp. on Inform. Th., Budapest, Hungary, p. 184, 24-28 Jun. 1991.

98. K.M. Cheung and A. Kiely, "An efficient variable length coding scheme for an iid source," Proc. Data Compression Conf., Snowbird, UT, 28-30 Mar. 1995, J.A. Storer and M. Cohn, eds., pp. 182-191, IEEE Computer Society Press, Los Alamitos, CA, 1995.

99. P. Elias, "Universal codeword sets and representations of the integers," IEEE Trans. on Inform. Th., Vol. IT-21, No. 2, pp. 194-203, Mar. 1975.

100. Q.F. Stout, "Searching and encoding for infinite ordered sets," Intl. J. of Computer and Inform. Sciences, Vol. 11, No. 1, pp. 55-72, 1982.

101. D.E. Knuth, "Supernatural numbers," The Mathematical Gardner, D. A. Klarner, ed., Wadsworth, Belmont, CA, pp. 310-325, 1981.

102. R. Ahlswede, T.S. Han, and K. Kobayashi, "Universal coding of integers and unbounded search trees," IEEE Trans. on Inform. Th., Vol. 43, No. 2, pp. 669-682, Mar. 1997.

103. R.M. Capocelli and A. De Santis, "Regular universal codeword sets," IEEE Trans. on Inform. Th., Vol. IT-32, No. 1, pp. 129-133, Jan. 1986.

104. A. Apostolico and A.S. Fraenkel, "Robert transmission of unbounded strings using Fibonacci representations," IEEE Trans. on Inform. Th., Vol. IT-33, No. 2, pp. 238-245, Mar. 1987.

105. R.M. Capocelli, "Comments and additions to 'Robust transmission of unbounded strings using Fibonacci representations," IEEE Trans. on Inform. Th., Vol. 35, No. 1, pp. 191-193, Jan. 1989.

106. K.B. Lakshmanan, "On univeral codeword sets," IEEE Trans. on Inform. Th., Vol. IT-27, No. 5, pp. 659-662, Sep. 1981.

107. M. Wang, "Almost asymptotically optimal flag encoding of the integers," IEEETrans. on Inform. Th., Vol. 34, No. 2, pp. 324-326, Mar. 1988.

108. Q. Stout, "Improved prefix encodings of the natural numbers," IEEE Trans. on Inform. Th., Vol. IT-26, No. 5, pp. 607-609, Sep. 1980.

109. H. Yamamoto and H. Ochi, "A new asymptotically optimal code for the positive integers," IEEE Trans. on Inform. Th., Vol. 37, No. 5, pp. 1421-1429, Sep. 1991.

110.E.M. Reingold and X. Shen, "More nearly optimal algorithms for unbounded searching, part I: the finite case [and] part II: the transfinite case," SIAM J. Comput., Vol. 20, No. 1, pp. 156-208, Feb. 1991.

111.J.L. Bentley and A.C.-C. Yao, "An almost optimal algorithm for unbounded searching," Inform. Processing Letters, Vol. 5, No. 3, pp. 82-87, Aug. 1976.

112.A.S. Fraenkel, "Systems of numeration," American Math. Monthly, Vol. 92, No. 2, pp. 105-114, 1985.

113.A. Moffat and J. Zobel, "Parameterised compression for sparse bitmaps," Proc.of 15th Annual Intl. ACM SIGIR Conf. on Research and Development in Inform. Retrieval, Copenhagen, Denmark, SIGIR Forum, N. Belkin, P. Inguersen, and A. M. Pejtersen, eds., pp. 274-285, 21-24 Jun. 1992.

114.G. Linoff and C. Stanfill, "Compression of indexes with full positional information in very large text databases," Proc. of 16th Annual Intl. ACM SIGIR Conf. on Research and Development in Inform. Retrieval, Pittsburgh, PA, SIGIR Forum, pp. 88-95, Jun. 1993.

115.A.S. Fraenkel and S.T. Klein, "Novel compression of sparse bit-strings, preliminary report," Combinatorial Algorithms on Words, Nato ASI Series, Vol. F12, A. Apostolico and Z. Galil, eds., pp. 170-183, Springer Verlag, New York, 1985.

116.T. Amemiya and H. Yamamoto, "A new class of the universal representation for the positive integers," IEICE Trans. Fundamentals, Vol. E76-A, No. 3, pp. 447-452, Mar. 1993.

117.T.C. Hu and A.C. Tucker, "Optimal computer search trees and variable-length alphabetical codes," SIAM J. Appl. Math., Vol. 21, No. 4. pp. 514-523, Dec. 1971.

118.A.M. Garsia and M.L. Wachs, "A new algorithm for minimum cost binary trees," SIAM J. Comput., Vol. 6, No. 4, pp. 622-642, Dec. 1977.

119.M. Klawe and B. Mumey, "Upper and lower bounds on constructing alphabetic binary trees," Proc. of the Fourth Annual ACM-SIAM Symp. on Discrete Algorithms, Austin, TX, pp. 185-193, 25-27 Jan. 1993.

120.T.C. Hu and J.D. Morgenthaler, "Optimum alphabetic binary trees," Combinatorics and Computer Science, CCS'95, 8th Franco-Japanese and 4th Franco-Chinese Conf., Brest, France, Jul. 1995, Lecture Notes in Computer Science, Vol. 1120, M. Deza, R. Euler, and I. Manoussakis, eds., pp. 234-243, Springer Verlag, New York, 1996.

121.J. Pradhan and C.V. Sastry, "A new version of Hu-Tucker algorithm," Proc. of the 29th Annual Convention of the Computer Society of India, CSI-94, Calcutta, India, pp. 27-31, 1994.

122.E.N. Gilbert and E.F. Moore, "Variable-length binary encodings," Bell System Technical Journal, Vol. 38, pp. 933-967, Jul. 1959.

123.R.W. Yeung, "Alphabetic codes revisited," IEEE Trans. on Inform. Th., Vol. 37, No. 3, pp. 564-572, May 1991.

124.N. Nakatsu, "Bounds on the redundancy of binary alphabetical codes," IEEE Trans. on Inform. Th., Vol. 37, No. 4, pp. 1225-1229, Jul. 1991.

125.D. Sheinwald, "On binary alphabetical codes," Proc. Data Compression Conf., Snowbird, UT, 1992, pp. 112-121, IEEE Computer Society Press, Los Alamitos, CA, 1992.

126.R. De Prisco and A. De Santis, "On binary search trees," Inform. Processing Letters, Vol. 45, No. 5, pp. 249-253, 2 Apr. 1993.

127.D.J. Kleitman and M.E. Saks, "Set orderings requiring costliest alphabetic binary trees," SIAM J. Alg. Disc. Meth., Vol. 2, No. 2, pp. 142-146, Jun. 1981.

128.T.C. Hu and K.C. Tan, "Least upperbound on the cost of optimum binary search trees," Acta Informatica, Vol. 1, pp. 307-310, 1972.

129.P. Ramanan, "Testing the optimality of alphabetic trees," Theoretical Computer Science, Vol. 93, pp. 279-301, 1992.

130.D.C. Van Voorhis, "Constructing codes with ordered codeword lengths," IEEETrans. on Inform. Th., Vol. 21, No. 1, pp. 105-106, Jun. 1975.

131.J. Abrahams, "Codes with monotonic codeword lengths," Inform. Processing and Management, Vol. 30, No. 6, pp. 759-764, 1994.

132.M.J. Lipman and J. Abrahams, "Minimum average cost testing for partially ordered components," IEEE Trans. on Inform. Th., Vol. 41, No. 1, pp. 287-291, Jan. 1995.

133.J. Abrahams, "Parallelized Huffman and Hu-Tucker searching," IEEE Trans. onInform. Th., Vol. 40, No. 2, pp. 508-510, Mar. 1994.

134.T.C. Hu, D.J. Kleitman, and J.K. Tamaki, "Binary trees optimum under various criteria," SIAM J. Appl. Math., Vol. 37, No. 2, pp. 246-256, Oct. 1979.

135.A. Moffat, A. Turpin, and J. Katajainen, "Space-efficient construction of optimal prefix codes," Proc. Data Compression Conf., Snowbird, UT, 28-30 Mar. 1995, J. A. Storer and M. Cohn, eds., pp. 192-201, IEEE Computer Society Press, Los Alamitos, CA, 1995.

136.J. Katajainen, A. Moffat and A. Turpin, "A fast and space-economical algorithm for length-limited coding," Proc. Intl. Symp. on Algorithms and Computation, Cairns, Australia, Dec. 1995, Lecture Notes in Computer Science, Vol. 1004, J. Staples, P. Eades, N. Katoh, and A. Moffat, eds., pp. 12-21, Springer Verlag, New York, 1995.

137.A. Turpin and A. Moffat, "Practical length-limited coding for large alphabets," Computer J., Vol. 38, No. 5, pp. 339-347, 1995.

138.L.L. Larmore and D.S. Hirschberg, "A fast algorithm for optimal length-limited Huffman codes," JACM, Vol. 37, No. 3, pp. 464-473, Jul. 1990.

139.M.R. Garey, "Optimal binary search trees with restricted maximal depth," SIAM J. Comput., Vol. 3, No. 2, pp. 101-110, Jun. 1974.

140.E.N. Gilbert, "Codes based on inaccurate source probabilities," IEEE Trans. onInform. Th., Vol. IT-17, No. 3, pp. 304-314, May 1971.

141.T.C. Hu and K.C. Tan, "Path length of binary search trees," SIAM J. Appl. Math., Vol. 22, No. 2, pp. 225-234, Mar. 1972.

142.D.C. Van Voorhis, "Constructing codes with bounded codeword lengths," IEEETrans. on Inform. Th., Vol. IT-20, No. 2, pp. 288-290, Mar. 1974.

143.H.Murakami, S. Matsumoto, and H. Yamamoto, "Algorithm for construction of variable length code with limited maximum word length," IEEE Trans. on Communications, Vol. COM-32, No. 10, pp. 1157-1159, Oct. 1984.

144.J.L.P.De Lameillieure, "A heuristic algorithm for the construction of a code with limited word length," IEEE Trans. on Inform. Th., Vol. IT-33, No. 3, pp. 438-443, May 1987; "Correction," IEEE Trans. on Inform. Th., Vol. 34, No. 4, pp. 893-894, Jul. 1988.

145.J. Pradhan and C.V. Sastry, "On optimal weighted binary trees," Intl. J. of High Speed Computing, Vol. 7, No. 3, pp. 445-464, 1995.

146.A. Itai, "Optimal alphabetic trees," SIAM J. Comput., Vol. 5, No. 1, pp. 9-18, Mar. 1976.

147.L.L. Larmore and T.M. Przytycka, "A fast algorithm for optimum height-limited alphabetic binary trees," SIAM J. Comput., Vol. 23, No. 6, pp. 1283-1312, Dec. 1994.

148.R.M. Capocelli and A. De Santis, "On the redundancy of optimal codes with limited word length," IEEE Trans. on Inform. Th., Vol. 38, No. 2, pp. 439-445, Mar. 1992.

149.Y. Chu, "An extended result of Kleitman and Saks concerning binary trees," DiscreteAppl. Math., Vol. 10, pp. 255-259, 1985.

150.S. Anily and R. Hassin, "Ranking the best binary trees," SIAM J. Comput., Vol. 18, No. 5, pp. 882-892, Oct. 1989.

151.D.S. Parker, Jr., "Conditions for optimality of the Huffman algorithm," SIAM J.Comput., Vol. 9, No. 3, pp. 470-489, Aug. 1980.

152.I.J. Taneja, "A short note on the redundancy of degree α," Inform. Sciences, Vol. 39, pp. 211-216, 1986.

153.U. Zwick, "An extension of Khrapchenko's theorem," Inform. Processing Letters, Vol. 37, pp. 215-217, 28 Feb. 1991.

154.P.A. Humblet, "Generalization of Huffman coding to minimize the probability of buffer overflow," IEEE Trans. on Inform. Th., Vol. IT-27, No. 2, pp. 230-232, Mar. 1981.

155.A.C. Blumer and R.J. McEliece, "The Rényi redundancy of generalized Huffman codes," IEEE Trans. on Inform. Th., Vol. 34, No. 5, pp. 1242-1249, Sep. 1988.

156.D.E. Knuth, "Huffman's algorithm via algebra," J. of Combinatorial Theory, Series A, Vol. 32, pp. 216-224, 1982.

157.E.S. Schwartz, "An optimum encoding with minimum longest code and total number of digits," Inform. and Control, Vol. 7, pp. 37-44, 1964.

158.L.T. Kou, "Minimum variance Huffman codes," SIAM J. Comput., Vol. 11, No. 1, pp. 138-148, Feb. 1982.

159.Y. Horibe, "Remarks on 'compact' Huffman trees," J. of Combinatorics, Inform. and System Sciences, Vol. 9, No. 2, pp. 117-120, 1984.

160.G. Markowsky, "Best Huffman trees," Acta Informatica, Vol. 16, pp. 363-370, 1981.

161.C.S. Chang and J.A. Thomas, "Huffman algebras for independent random variables," Discrete Event Dynamic Systems: Theory and Applications, Vol. 4, No. 1, pp. 23-40, Feb. 1994, and Proc. IEEE Intl. Symp. on Inform. Th., San Antonio, TX, p. 215, 17-22 Jan. 1993.

162.C.Q. Zhang, "Optimal alphabetic binary tree for a nonregular cost function," DiscreteAppl. Math., Vol. 8, pp. 307-312, 1984.

163.D.G. Kirkpatrick and M.M. Klawe, "Alphabetic minimax trees," SIAM J. Comput., Vol. 14, No. 3, pp. 514-526, Aug. 1985.

164.T.M. Cover, "On the competitive optimality of Huffman codes," IEEE Trans. on Inform. Th., Vol. 37, No. 1, pp. 172-174, Jan. 1991.

165.M. Feder, "A note on the competitive optimality of the Huffman code," IEEE Trans. on Inform. Th., Vol. 38, No. 2, pp. 436-439, Mar. 1992.

166.H. Yamamoto and T. Itoh, "Competitive optimality of source codes," IEEE Trans. on Inform. Th., Vol. 41, No. 6, pp. 2015-2019, Nov. 1995.

167.L.L. Larmore, "Minimum delay codes," SIAM J. Comput., Vol. 18, No. 1, pp. 82-94, Feb. 1989.

168.D. Cohen and M.L. Fredman, "Weighted binary trees for concurrent searching," J. of Algorithms, Vol. 20, pp. 87-112, 1996.

169.R.M. Karp, "Minimum-redundancy coding for the discrete noiseless channel," IRETrans. on Inform. Th., pp. 27-38, Jan. 1961.

170.M.J. Golin and G. Rote, "A dynamic programming algorithm for constructing optimal prefix-free codes for unequal letter costs," Proc. of the 22nd Intl. Colloquium on Automata, Languages, and Programming, ICALP 1995, Szeged, Hungary 10-14 Jul. 1995, Lecture Notes in Computer Science, Vol. 944, Z. Fulop, F. Gecseg, eds., pp. 256-267, Springer Verlag, New York, 1995.

171.R.M. Krause, "Channels which transmit letters of unequal duration," Inform. And Control, Vol. 5, pp. 13-24, 1962.

172.T. Kawabata and K. Kobayashi, "Improvement of upper bound to the average cost of the variable length code," Proc. of the IEEE Intl. Symp. on Inform. Th., Trondheim, Norway, 27 Jun.-1 Jul. 1994, p. 188.

173.L. Carter and J. Gill, "Conjectures on uniquely decipherable codes," IEEE Trans. on Inform. Th., Vol. IT-20, No. 3, pp. 394-396, May 1974.

174.R. De Prisco and G. Persiano, "Characteristic inequalities for binary trees," Inform.Processing Letters, Vol. 53, pp. 201-207, 1995.

175.E.N. Gilbert, "Coding with digits of unequal cost," IEEE Trans. on Inform. Th., Vol. 41, No. 2, pp. 596-600, Mar. 1995.

176.K. Mehlhorn, "An efficient algorithm for constructing nearly optimal prefix codes," IEEE Trans. on Inform. Th., Vol. IT-26, No. 5, pp. 513-517, Sep. 1980.

177.D.A. Altenkamp and K. Mehlhorn, "Codes: unequal probabilities, unequal letter costs," JACM, Vol. 27, No. 3, pp. 412-427, Jul. 1980.

178.D.M. Choy and C.K. Wong, "Construction of optimal α-β leaf trees with applications to prefix code and information retrieval," SIAM J. Comput., Vol. 12, No. 3, pp. 426-446, Aug. 1983.

179.Y.N. Patt, "Minimum search tree structures for data partitioned into pages," IEEETrans. on Computers, Vol. C-21, No. 9, pp. 961-967, Sep. 1972.

180.Y.N. Patt, "Variable length tree structures having minimum average search time," CACM, Vol. 12, No. 2, pp. 72-76, Feb. 1969.

181.T. Ottmann, A.L. Rosenberg, H.W. Six, and D. Wood, "Binary search trees with binary comparison cost," Intl. J. of Computer and Inform. Sciences, Vol. 13, No. 2, pp. 77-101, 1984.

182.B. Varn, "Optimal variable length codes (arbitrary symbol cost and equal code word probability)," Inform. and Control, Vol. 19, pp. 289-301, 1971.

183.Y. Perl, M.R. Garey, S. Even, "Efficient generation of optimal prefix code: equiprobable words using unequal cost letters," JACM, Vol. 22, No. 2, pp. 202-214, Apr. 1975.

184.S. Kapoor and E.M. Reingold, "Optimum lopsided binary trees," JACM, Vol. 36, No. 3, pp. 573-590, Jul. 1989.

185.N. Cot, "Characterization and design of optimal prefix codes," Ph. D. dissertation, Stanford University, Palo Alto, CA, 1977.

186.M.J. Golin and N. Young, "Prefix codes: equiprobable words, unequal letter costs," SIAM J. Comput., Vol. 25, No. 6, pp. 1281-1292, Dec. 1996.

187.S.N. Choi and M.J. Golin, "Lopsided trees: analysis, algorithms, and applications" (extended abstract), preprint, 11 Dec. 1994, golin@cs.ust.hk.

188.S.A. Savari, "Some notes on Varn coding," IEEE Trans. on Inform. Th., Vol. 40, No. 1, pp. 181-186, Jan. 1994.

189.C.V. Sastry and J. Pradhan, "Lower bounds to the external path length of a lopsided binary tree," Intl. J. of High Speed Computing, Vol. 4, No. 3, pp. 169-178, 1992.

190.Y. Horibe, "Notes on Fibonacci trees and their optimality," Fibonacci Quarterly, Vol. 21, No. 2, pp. 118-128, 1983.

191.D.K. Chang, "On Fibonacci k-ary trees," Fibonacci Quarterly, Vol. 24, No. 3, pp. 258-262, 1986.

192.J. Abrahams, "Varn codes and generalized Fibonacci trees," Fibonacci Quarterly, Vol. 33, No. 1, pp. 21-25, Feb. 1995.

193.J. Abrahams, "Nonexhaustive generalized Fibonacci trees in unequal costs coding problems," Fibonacci Quarterly, submitted, and "Minimizing the maximum codeword cost," Proc. of the IEEE Intl. Symp. on Inform. Th., Whistler, BC, p. 326, 17-22 Sep. 1995.

194.K. Hinderer, "On dichotomous search with direction-dependent costs for a uniformly hidden object," Optimization, Vol. 21, No. 2, pp. 215-229, 1990.

195.Y. Horibe, "On dichotomous complexity of the Fibonacci Tree," Applications of Fibonacci Numbers, Vol. 6, Pullman, WA, 1994, G. E. Bergum et al., eds., pp. 251-256, Kluwer, the Netherlands, 1996.

196.Y. Horibe, "Balance morphology of a binary tree," Applications of Fibonacci Numbers, Vol. 5, G.E. Bergum et al., eds., pp. 345-354, Kluwer, the Netherlands, 1993.

197.Y. Horibe, "A Fibonacci theme on balanced binary trees," Fibonacci Quarterly, Vol. 30, No. 3, pp. 244-250, Aug. 1992.

198.Y. Horibe, "An entropy view of Fibonacci trees," Fibonacci Quarterly, Vol. 20, pp. 168-178, May 1982.

199.Y. Horibe, "Entropy of terminal distributions and the Fibonacci trees," Fibonacci Quarterly, Vol. 26, No. 2, pp. 135-140, May 1988.

200.J. Abrahams and M.J. Lipman, "Zero-redundancy coding for unequal code symbol costs," IEEE Trans. on Inform. Th., Vol. 38, No. 5, pp. 1583-1586, Sep. 1992.

201.C.E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, Jul., Oct. 1948.

202.K.A. Schouhamer Immink, Coding Techniques for Digital Recorders, Prentice Hall, New York, 1991.

203.I. Csiszár, "Simple proofs of some theorems on noiseless channels," Inform. andControl, Vol. 14, pp. 285-298, 1969.

204.K. Kobayashi, "On coding theorems with modified length functions," Logic, Language, and Computation, Lecture Notes in Computer Science, Vol. 792, pp. 255-259, Springer Verlag, New York, 1994.

205.K. Kobayashi, "The Kolmogorov complexity, universal distribution, and coding theorem for generalized length functions," IEEE Trans. on Inform. Th., Vol. 43, No. 3, pp. 816-826, May 1997.

206.J. Abrahams, "Coupled sequences of generalized Fibonacci trees and unequal costs coding problems," Fibonacci Quarterly, to appear, Nov. 1997.

207.A.S. Fraenkel and S.T. Klein, "Bidirectional Huffman coding," Computer J., Vol. 33, No. 4, pp. 296-307, 1990.

208.D. Gillman and R.L. Rivest, "Complete variable-length 'fix-free' codes," Designs, Codes and Cryptography, Vol. 5, pp. 109-114, 1995.

209.Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes," IEEETrans. on Communications, Vol. 43, No. 2/3/4, pp. 158-162, Feb./Mar./Apr. 1995.

210.R. Ahlswede, B. Balkenhol, and L. Khachatrian, "Some properties of fix-free codes," preprint 97-039, Diskrete Strukturen in der Mathematik, Universität Bielefeld.

211.T.J. Ferguson and J.H. Rabinowitz, "Self-synchronizing Huffman codes," IEEE Trans. on Inform. Th., Vol. IT-30, No. 4, pp. 687-693, Jul. 1984.

212.R.M. Capocelli, A. De Santis, L. Gargano and U. Vaccaro, "On the construction of statistically synchronizable codes," IEEE Trans. on Inform. Th., Vol. 38, No. 2, pp. 407-414, Mar. 1992.

213.B.L. Montgomery and J. Abrahams, "Synchronization of binary source codes," IEEETrans. on Inform. Th., Vol. IT-32, No. 6, pp. 849-854, Nov. 1986.

214.A.E. Escott and S. Perkins, "The construction of binary Huffman codes with two short synchronizing codewords," Proc. of the 1996 Intl. Symp. on Inform. Th. and Its Applications, Victoria, BC, pp. 294-297, 17-20 Sep. 1996.

215.A.E. Escott and S. Perkins, "Constructing good binary synchronous Huffman codes," Proc. 1995 Intl. Symp. on Synchronization, Saalbau, Essen, Germany, K. A. Schouhamer Immink and A. J. Vinck, eds., pp. 105-110, 14-15 Dec. 1995.

216.B. Rudner, "Construction of minimum redundancy codes with an optimum synchronizing property," IEEE Trans. on Inform. Th., Vol. IT-17, No. 4, pp. 478-487, Jul. 1971.

217.T. Berger and R.W. Yeung, "Optimium '1'-ended binary prefix codes," IEEE Trans. on Inform. Th., Vol. 36, No. 6, pp. 1435-1441, Nov. 1990.

218.W.M. Lam and S.R. Kulkarni, "Extended synchronizing codewords for binary prefix codes," IEEE Trans. on Inform. Th., Vol. 42, No. 3, pp. 984-987, May 1996.

219.S.M. Lei, "The construction of efficient variable-length codes with clear synchronizing codewords for digital video applications," Proc. Conf. on Visual Communications and Image Processing, Boston, MA, SPIE Vol. 1605, K.H. Tzou and T. Koga, eds., pp. 863-873, 11-13 Nov. 1991

220.M.R. Titchener, "Digital encoding by means of new T-codes to provide improved data synchronisation and message integrity," IEE Proc., Pt. E, Computers and Digital Techniques, Vol. 131, No. 4, pp. 151-153, Jul. 1984.

221.M.R. Titchener, "Construction and properties of the augmented and binary-depletion codes," IEE Proc., Pt. E, Computers and Digital Techniques, Vol. 132, No. 3, pp. 163-169, May 1985.

222.M.R. Titchener, "The synchronization of variable-length codes," IEEE Trans. on Inform. Th., Vol. 43, No. 2, pp. 683-691, Mar. 1997.

223.G.R. Higgie and A.G. Williamson, "Properties of low augmentation level T-codes," IEE Proc., Pt. E, Computers and Digital Techniques, Vol. 137, No. 2, pp. 129-132, Mar. 1990.

224.G.R. Higgie, "Self synchronizing T-codes to replace Huffman codes," Proc. of the IEEE Intl. Symp. on Inform. Th., San Antonio, TX, p. 336, 17-22 Jan. 1993.

225.G.R. Higgie, "Database of best T-codes," IEE Proc.-Comput. Digit. Tech., Vol. 143, No. 4, pp. 213-218, Jul. 1996.

226.P.F. Swaszek and P. Willett, "On the design of T-codes," Proc. Conf. on Inform.Sciences and Systems, Johns Hopkins University, Baltimore, MD, p. 93, Mar. 1995.

227.J.C. Maxted and J.P. Robinson, "Error recovery for variable length codes," IEEE Trans. on Inform. Th., Vol. IT-31, No. 6, pp. 794-801, Nov. 1985.

228.P.F. Swaszek and P. DiCicco, "More on the error recovery for variable-length codes," IEEE Trans. on Inform. Th., Vol. 41, No. 6, pp. 2064-2071, Nov. 1995.

229.D.W. Gillman, M. Mohtashemi, and R.L. Rivest, "On breaking a Huffman code," IEEE Trans. on Inform. Th., Vol. 42, No. 3, pp. 972-976, May 1996.

230.F. Jelinek and G. Longo, "Algorithms for source coding," Coding and Complexity, G. Longo, ed., Intl. Centre for Mechanical Sciences-Courses and Lectures No. 216, Springer Verlag, New York, pp. 293-330, 1975.

231.J. Verhoeff, "A new data compression technique," Annals of Systems Research, Vol. 6, pp. 139-148, 1977.

232.P.R. Stubley and I.F. Blake, "On a discrete probability distribution matching problem," preprint, 16 Nov. 1992.

233.J.P.M. Schalkwijk, F. Antonio, and R. Petry, "An efficient algorithm for data reduction," Proc. Fifth Hawaii Intl. Conf. on System Sciences, pp. 498-499, 1972.

234.J.P.M. Schalkwijk, "On Petry's extension of a source coding algorithm," Proc. Second Symp. Inform. Th. Benelux, Zoetermeer, the Netherlands, pp. 99-102, 21-22 May 1981.

235.F. Fabris, A. Sgarro, and R. Pauletti, "Tunstall adaptive coding and miscoding," IEEETrans. on Inform. Th., Vol. 42, No. 6, pp. 2167-2180, Nov. 1996.

236.T. Algra, "Fast and efficient variable-to-fixed-length coding algorithm," Electronics Letter, Vol. 28, No. 15, pp. 1399-1401, 16 Jul. 1992.

237.A. Lempel, S. Even, and M. Cohn, "An algorithm for optimal prefix parsing of a noiseless and memoryless channel," IEEE Trans. on Inform. Th., Vol. IT-19, No. 2, pp. 208-214, Mar. 1973.

238.F. Jelinek and K.S. Schneider, "On variable-length-to-block coding," IEEE Trans. on Inform. Th., Vol. IT-18, No. 6, pp. 765-774, Nov. 1972.

239.S. Kapoor and E.M. Reingold, "Recurrence relations based on minimization and maximization," J. of Math. Analysis and Applications, Vol. 109, pp. 591-604, 1985.

240.J. Abrahams, "Correspondences between variable length parsing and coding problems," Proc. IMA Workshop on the Math. of Inform. Coding, Extraction, and Distribution, Minneapolis, MN, 11-15 Nov. 1996, G. Cybenko, D. O'Leary, and J. Rissanen, eds., Springer Verlag, New York, to appear.

241.F. Fabris, "Variable-length-to-variable-length source coding: a greedy step-by-step algorithm," IEEE Trans. on Inform. Th., Vol. 38, No. 5, pp. 1609-1617, Sep. 1992.

242.G.H. Freeman, "Divergence and the construction of variable-to-variable-length lossless codes by source-word extensions," Proc. Data Compression Conf., Snowbird, UT, 30 Mar. - 2 Apr. 1993, J. A. Storer and M. Cohn, eds., pp. 79-88, IEEE Computer Society Press, Los Alamitos, CA, 1993.

243.P.R. Stubley, "Adaptive variable-to-variable length codes," Proc. Data Compression Conf., Snowbird, UT, 29-31 Mar. 1994, J. A. Storer and M. Cohn, eds., pp. 98-105, IEEE Computer Society Press, Los Alamitos, CA, 1994.

244.D.E. Knuth and A.C. Yao, "The complexity of nonuniform random number generation," Algorithms and Complexity: New Directions and Recent Results, Carnegie Mellon University, Pittsburgh, PA, 7-9 Apr. 1976, J.F. Traub, ed., pp. 357-428, Academic, New York, 1976.

245.J. Abrahams, "Generation of discrete distributions from biased coins," IEEE Trans. on Inform. Th., Vol. 42, No. 5, pp. 1541-1546, Sep. 1996.

246.J. Katajainen and E. Mäkinen, "Tree compression and optimization with applications," Intl. J. of Foundations of Computer Science, Vol. 1, No. 4, pp. 425-447, 1990.

247.E. Mäkinen, "A survey on binary tree codings," Computer J., Vol. 34, No. 5, pp. 438-443, 1991.

248.E.S. Schwartz and B. Kallick, "Generating a canonical prefix encoding," CACM, Vol. 7, No. 3, pp. 166-169, Mar. 1964.

249.D.S. Hirschberg and D.A. Lelewer, "Efficient decoding of prefix codes," CACM, Vol. 33, No. 4, pp. 449-459, Apr. 1990.

250.R. Hashemian, "Memory efficient and high-speed search Huffman coding," IEEE Trans. on Communications, Vol. 43, No. 10, pp. 2576-2581, Oct. 1995.

251.H. Park and V.K. Prasanna, "Area efficient VLSI architectures for Huffman coding," IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 40, No. 9, pp. 568-575, Sep. 1993.

252.J. Xianchun, "Prefix code translation by mapping," J. of Computer Science andTechnology, Vol. 9, No. 2, pp. 175-181, 1994.

253.D.R. McIntyre and F.G. Wolff, "An efficient implementation of Huffman decode tables," Congressus Numerantium, Vol. 91, pp. 79-92, 1992.

254.D.S. Parker and P. Ram, "The construction of Huffman codes is a submodular ('convex') optimization problem over a lattice of binary trees," SIAM J. Comput., submitted.

255.K. Kobayashi and T.S. Han, "On the pre-order coding for complete k-ary coding trees," Proc. of the 1996 Intl. Symp. on Inform. Th. and Its Applications, Victoria, BC, pp. 302-303, 17-20 Sep. 1996.

256.J.N. Franklin and S.W. Golomb, "A function-theoretic approach to the study of nonlinear recurring sequences," Pacific J. of Math., Vol. 56, No. 2, pp. 455-468, 1975.

257.S. Even and A. Lempel, "Generation and enumeration of all solutions of the characteristic sum condition," Inform. and Control, Vol. 21, pp. 476-482, 1972.

258.D.W. Boyd, "The asymptotic number of solutions of a Diophantine equation from coding theory," J. of Combinatorial Theory (A), Vol. 18, pp. 210-215, 1975.

259.E. Norwood, "The number of different possible compact codes," IEEE Trans. on Inform. Th., Vol. IT-13, pp. 613-616, Oct. 1967.

260.A. Rényi, "On the enumeration of search-codes," Acta Math. Acad. Sci. Hung., Vol. 21, pp. 27-33, 1970, and Selected Papers of Alfréd Rényi, Vol. 3, P. Turán, ed., Akadémiai Kiadó, Budapest, Hungary, 1976.

261.J. Komlós, W. Moser, and T. Nemetz, "On the asymptotic number of prefix codes," Mitteilungen aus dem Math. Seminar Giessen, Vol. 165, Coxeter-Festschrift, Part III, pp. 35-48, 1984.

262.I.F. Blake, G.H. Freeman, and P.R. Stubley, "On the enumeration of dyadic distributions," Sequences II, Methods in Communication, Security, and Computer Science, R. Capocelli, A. De Santis, and U. Vaccaro, eds., Springer Verlag, New York, 1993, pp. 3-11.

263.L.S. Bobrow and S.L. Hakimi, "Graph theoretic prefix codes and their synchronizing properties," Inform. and Control, Vol. 15, pp. 70-94, 1969.

264.B. Fitingof and Z. Waksman, "Fused trees and some new approaches to source coding," IEEE Trans. on Inform. Th., Vol. 34, No. 3, pp. 417-424, May 1988.