CODING FOR GENERAL PENALTIES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Michael B. Baer

June 2003

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Thomas M Cover

Thomas M. Cover
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

John T. Gill III

John T. Gill, III

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Benjamin Van Roy

Approved for the University Committee on Graduate Studies.

Abstract

Huffman coding finds a prefix-free code that minimizes the expected codeword length for a probability mass function. However, there are many practical situations in which the constraints and goals may be different from the linear penalty of minimizing expected length. For example, we may wish to avoid long codewords in some situations, since these may cause an excessive wait for transmission of unlikely events. Therefore, we may desire a nonlinear penalty in the codeword lengths. Such penalties arise naturally in group testing for diagnostic systems, queueing, remote exploration, and military intelligence.

We examine families of coding problems with nonlinear penalties. These include generalizations of Huffman coding suitable for communication and analysis, extending instances of source coding found in the literature. Alternative penalty measures previously considered include exponential average in [15, 18, 39, 45, 53, 55, 78, 87] and maximal pointwise redundancy in [24]. We find a novel framework encompassing these two penalties, as well as the linear and other natural penalties, resulting in properties and algorithms suitable for all of them.

We next turn to generalized quasilinear convex penalties; these are penalties that can be formulated as $\sum_i f(l_i, p_i)$, where l_i denotes the length of the ith codeword, p_i denotes the corresponding probability, and f is convex and increasing in l_i. This class of penalties includes several previously proposed penalties, among them a general convex problem proposed by Campbell in [15]. We give an efficient algorithm that, for Campbell's problem, performs the optimization in quadratic time and linear space; we also present penalty bounds for this solution. Finally, we consider coding for infinite alphabets, a problem not fully understood even for the linear penalty.

Acknowledgments

The road has been a long and interesting one, and I am indebted to several fellow travelers, mentors, helpers, and companions. The Alma M. Collins Fellowship, a Stanford Graduate Fellowship, gave me the freedom to find my own way. Without this support, this thesis would not have been possible. Support from the National Science Foundation and the Multidisciplinary University Research Initiative was also instrumental in funding my research.

The Information Theory Group has provided the ideal environment for living life and engaging in academic thought, punctuated by conversations about sports, movies, and gambling. In addition to Professor Cover, I should thank to those who came before I did (Suhas Diggavi, Paul Fahn, Yiannis Kontoyiannis, Assaf Zeevi, and Arak Sutivong) and those who came after (Young-Han Kim, Styrmir Sigurjonsson, and George Gemelos). But I am especially thankful to my contemporaries, Mung Chiang, Jon Yard, Joshua Sweetkind-Singer, and particularly David Julian. They have all provided help in the form of everything from references to proofreading and research thoughts to word choice.

In addition to these colleagues are those teachers and mentors who were of special help. Stephen Boyd was especially helpful and patient regarding matters of convexity. Literature help was provided by Donald Knuth of Stanford, Wojciech Szpankowski of Purdue, George Turin of Berkeley, D. Stott Parker of UCLA, Mordecai Golin of the Hong Kong University of Science and Technology, and Alistair Moffat of the University of Melbourne. Additional literature pointers were provided by Hewlett-Packard Labs' Gadiel Seroussi and Marcelo Weinberger, who were also my mentors during my extended internship at HPL. They sparked an interest in source coding, security, and Rényi entropy, which has continued throughout my doctoral research.

I'd like to extend special thanks to T. C. Hu, who was able to comment on an early draft of my work, and to Andrew Brzezinski, my former roommate, for providing feedback at many points in my journey.

I also wish to thank my parents for their support and for only occasionally asking me when I was going to graduate. I want to thank friends and loved ones for their support as well.

Last but not least, I would like to thank my thesis readers as well as my orals chairman. Mine is the first Stanford oral session Joseph Kahn has chaired, and I chose him to do so because he taught the class that was my introduction to the field, the introductory class in information systems at UC Berkeley. Benjamin Van Roy came on relatively recently and has been a warm and receptive audience, with perceptive and thought-provoking views of my research. My co-advisor, John Gill, has been a great resource on matters regarding classical Huffman coding and the finer points of writing. Finally, great thanks is owed to my advisor, Thomas Cover, for the opportunity, the support, and the freedom necessary for this research. The spirit he imparts onto a diverse and eclectic research group is evident any time we meet, and his much-noted curiosity about aesthetically pleasing problems — as well as his innovation in finding solutions — inspired this work.

Notation

Notation	Meaning
$\in$	a member of
$\subseteq$	a subset of
$\subset$	a strict subset of
$\triangleq$	is defined as
$X\backslash Y$	set X excluding those elements in Y
$[x, y)$	the interval from x to y, including x but excluding y
$\lfloor\cdot\rfloor$	the largest integer at most $\cdot$
$\lceil\cdot\rceil$	the smallest integer at least $\cdot$
$\langle\cdot\rangle$	the fractional part of $\cdot$, $(\cdot) - \lfloor\cdot\rfloor$
$S = \{x, y, \ldots\}$	a set
$\|S\|$	the number of items (possibly $+\infty$) in set S
$c \cdot S$	the set consisting of cs for each $s \in S$
$-S$	the set consisting of $-s$ for each $s \in S$
$\{\cdot \mid A(\cdot)\}$	the set of $\cdot$ such that $A(\cdot)$ is satisfied
$\boldsymbol{v} = (v_1, v_2, \ldots)$	a (possibly infinite-element) vector
$v(i), v_i$	the ith element of vector v
$c \cdot \boldsymbol{v}$	constant c multiplied by each element of vector $\boldsymbol{v}$
$\boldsymbol{v} \times \boldsymbol{w}$	all possible products of elements of $\boldsymbol{v}$ and $\boldsymbol{w}$
$\boldsymbol{v} \succeq \boldsymbol{w}$	$v_i \geq w_i \ \forall\ i$
x^+	$\max(0, x)$

x^-	$\min(0, x)$
$\tilde{x}$	a modified version of x
x_{max}	a maximum value for x
x_{min}	a minimum value for x
x^*	optimal value for x
$x^\dagger$, $x^\ddagger$	optimal value for x with relaxed constraints
f^{-1}	the inverse function of f
$\cup_{i=x}^{y} S(i)$	the union of sets $S(x), S(x+1), \ldots,$ and $S(y)$
$\emptyset$	the empty set
$\mathbf{0}$	a vector of all zeroes (dimension clear from context)
0^k	a string of k zeroes
$\{0,1\}^*$	the set of all finite length binary sequences
$\{0,1\}^k$	the set of all length k binary sequences
1_τ	1 if τ is true, 0 otherwise
2^S	the set consisting of 2^s for each $s \in S$
$\forall$	for all (meaningful instances of)
$\exists$	there exists (a meaningful instance of)
Γ_n^0	the set of probability distributions over n items, none with probability zero
$\zeta(\alpha)$	the Zeta function, $\sum_{k=1}^{+\infty} k^{-\alpha}$
$\cdot = \Theta(X(b))$	$\lim_{b \to c} \frac{\lvert\cdot\rvert}{\lvert X(b)\rvert}$ exists, is nontrivial, and is finite. (The value of c, usually $+\infty$ though often $-\infty$ or 0, is always implied by the context.)
Λ	a null value
$\prod_{i=x}^{y} f(i)$	product of $f(i)$ from $i = x$ to y
$\prod_i f(i)$	product of $f(i)$ for all meaningful values of i
$\sum_{i=x}^{y} f(i)$	sum of $f(i)$ from $i = x$ to y
$\sum_{i \in \mathcal{X}} f(i)$	sum of $f(i)$ for all $i \in \mathcal{X}$

$\sum_i f(i)$	sum of $f(i)$ for all meaningful values of i
Φ	the Golden Mean, $\frac{\sqrt{5}+1}{2}$
$\arg\max_i X(i)$	value of i for which $X(i)$ is maximized
$\arg\min_i X(i)$	value of i for which $X(i)$ is minimized
$c(i), c_i$	the ith codeword
$\mathcal{C}^n$	the set of functions for which derivatives up to the nth exist and are continuous
$C_{\mathcal{X}}$	a code for alphabet $\mathcal{X}$
$\mathrm{CAT}_{i=x}^{y}\, c(i)$	concatenation of $c(x), c(x+1), \ldots,$ and $c(y)$
$D(\boldsymbol{p} \parallel \boldsymbol{q})$	Kullback Leibler divergence $\left[\sum_k p_k \lg \frac{p_k}{q_k}\right]$
$D_\alpha(\boldsymbol{p} \parallel \boldsymbol{q})$	Rényi divergence $\left[\frac{1}{\alpha-1} \lg \left(\sum_k \frac{p_k^\alpha}{q_k^{\alpha-1}}\right)\right]$
e	base of the natural logarithm, $e = 2.7182818284\ldots$
$E_{\boldsymbol{p}}[f(X)]$	The expected value of $f(X)$ under $\boldsymbol{p}$, $\sum_i p_i f(i)$
$H(\boldsymbol{p})$	Shannon entropy $[-\sum_k p_k \lg p_k]$
$H_\alpha(\boldsymbol{p})$	Rényi (α-)entropy $\left[\frac{1}{1-\alpha} \lg \sum_k p_k^\alpha\right]$
$H(\boldsymbol{p}, f)$	Generalized entropy for function f
iff	if and only if
$\inf_{i \in B} X(i)$	the value for which no $X(i)$ is smaller but there is an $X(i)$ arbitrarily close ($i \in B$)
$l(i), l_i$	the length of the ith codeword
$L_{\mathcal{X}}$	the codeword lengths of $C_{\mathcal{X}}$
$L'_{\mathcal{X}} \geq L_{\mathcal{X}}$	$L'_{\mathcal{X}}$ is at least $L_{\mathcal{X}}$ in terms of lexicographical order (with lengths sorted in nonincreasing order)
$L_{\mathcal{X}} + c$	constant c added to each element of vector $L_{\mathcal{X}}$
$L_{\mathcal{X}} + L'_{\tilde{\mathcal{X}}}$	the lengths of the tree created by having $L_{\mathcal{X}}$ as the subtree of one child of the root and $L'_{\tilde{\mathcal{X}}}$ as the subtree of the other
$l_i^\dagger(b, \boldsymbol{p})$	ideal codeword length $\left[-\frac{1}{1+b} \lg p_i + \lg \left(\sum_j p_j^{\frac{1}{1+b}}\right)\right]$

$\lg n$	logarithm of n base 2
$\lim_{i \uparrow c} X(i)$	limit of c approaching from below
$\lim_{i \downarrow c} X(i)$	limit of c approaching from above
$\lim_{i \to c} X(i)$	limit of c ($\uparrow$ if $c = +\infty$, $\downarrow$ if $c = -\infty$)
$\liminf_{i \to +\infty} X(i)$	$\lim_{c \to +\infty} \{\inf_{i \geq c} X(i)\}$
$\limsup_{i \to +\infty} X(i)$	$\lim_{c \to +\infty} \{-\inf_{i \geq c}[-X(i)]\}$
$\ln n$	natural logarithm of n (base e)
$\log_a n$	logarithm of n base a
$\log n$	logarithm of n in any base
m_i	the ith item
$\max(a, b)$	the maximum of a and b
$\max_i X(i)$	the maximum value of $X(i)$ (assumes this exists)
$\min(a, b)$	the minimum of a and b
$\min_i X(i)$	the minimum value of $X(i)$ (assumes this exists)
$\min^+_{i,j} \gamma_{i,j}$	the minimum strictly positive value of $\gamma_{i,j}$
n	size of the current problem, usually $\lvert\mathcal{X}\rvert$
$\mathbb{N}$	$\{1, 2, 3, \ldots\}$
N_k	$\{0, 1\}^k \backslash 0^k$
$N_S(i)$	the number of occurrences of i in sequence S
$\cdot = O(X(b))$	$\lim_{b \to c} \frac{\lvert \cdot \rvert}{\lvert X(b) \rvert}$ exists and is finite.
	(The value of c, usually $+\infty$ though often $-\infty$ or 0,
	is always implied by the context.)
$P\{A\}$	the probability of A
$P_X\{A(X)\}$	the probability of A (a function of random variable X)
$\boldsymbol{p}, \boldsymbol{q}$	probability vectors (elements sum to 1)
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^+$	the set of nonnegative real numbers
$\mathbb{R}^n$	the set of real-numbered n-vectors

$\mathbb{R}^{+\infty}$	the set of real-numbered countable-element vectors
$\mathcal{S}(\cdot)$	$\{1, 2, \ldots, \cdot\}$
$\operatorname{sgn}(x)$	$\begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$
$\mathcal{T}(\cdot)$	$\mathbb{N} \backslash \mathcal{S}(\cdot) = \{(\cdot) + 1, (\cdot) + 2, \ldots\}$
$\boldsymbol{v}, \boldsymbol{w}$	weight vectors (elements do not necessarily sum to 1)
$\mathcal{X}$	an alphabet
$\mathbb{Z}$	the set of integers, $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$

Contents

Abstract v

Acknowledgments vi

Notation viii

1 Introduction 1

1.1 An introductory problem 1

1.2 Motivation 2

1.3 Formalization 3

1.4 Prior work 9

1.5 Organization 11

2 Exponential penalties 12

2.1 Nonlinear penalties 12

2.2 An exponential penalty 15

2.2.1 Motivation 15

2.2.2 Real-valued problem solution and bounds 17

2.2.3 Integer-valued problem solution and algorithm 19

2.3 Properties of the exponential penalty 26

2.3.1 Sibling property 26

2.3.2 Interval properties 30

2.4 Another exponential penalty 33

2.4.1 Motivation 33

2.4.2 Problem structure and solution 34
2.5 Properties of both exponential penalties 36
2.5.1 Sibling property 36
2.5.2 Properties on the real line 37
2.5.3 Interval properties 40
2.6 Extending to other penalties 45

3 Redundancy penalties 48

3.1 Motivation, method, and properties 48
3.1.1 *d*-average *b*-redundancy (DABR) 48
3.1.2 Special cases 52
3.1.3 Bounds 54
3.2 Minimizing maximal pointwise redundancy 57
3.2.1 Minimax redundancy as a generalized Shannon code 58
3.2.2 Minimax redundancy as minimal DABR 59
3.2.3 Linear time methods for minimax redundancy 62

4 Generalized quasilinear convex penalties 67

4.1 Introduction and motivation 67
4.2 Bounds 69
4.3 The Coin Collector's problem 73
4.3.1 Problem statement and properties 73
4.3.2 Package-Merge algorithm and proof of correctness 74
4.4 Overall algorithm 75
4.5 A linear space algorithm 80
4.6 Further algorithmic optimization 84
4.7 Examples 85
4.8 Extending to more exotic penalties 86
4.9 Review of finite alphabet coding cases 88

5 Extensions to infinite alphabets 89

5.1 Motivation 89

5.2 Properties of optimal infinite item codes 92
5.3 Extending to other penalties 97
5.3.1 Existence of optimal codes 97
5.3.2 Construction of optimal codes 103

6 Concluding Remarks **106**
6.1 Summary of contributions 106
6.2 Extensions and future directions 107

Bibliography **109**

List of Tables

1.1 Examples of penalty functions 8

List of Figures

1.1 A five-codeword prefix-free code $C_{\mathcal{X}}$ represented by a binary tree . . . 5
1.2 Penalty classes we consider in this dissertation 11

2.1 Representations of a prefix-free code as a tree and as subintervals of $[0,1)$ 13
2.2 Huffman coding, $\boldsymbol{p} = (0.36, 0.30, 0.20, 0.14)$ 23
2.3 Exponential Huffman coding, $\boldsymbol{p} = (0.36, 0.30, 0.20, 0.14)$ and $a = 1.1$. 24
2.4 Tree representation of a partition of the alphabet $\mathcal{X}$ 27
2.5 Three optimal coding trees, $\boldsymbol{p} = (\frac{3}{8}, \frac{3}{16}, \frac{1}{8}, \frac{3}{32}, \frac{1}{16}, \frac{3}{64}, \frac{1}{32}, \frac{1}{32}, \frac{3}{128}, \frac{3}{128})$ . . . 29
2.6 Bottom-merging versus top-merging 32
2.7 Optimal code trees for two different distributions over the range of $b \in \mathbb{R}$ 39

3.1 Parameter space for minimal DABR coding with notable special cases 53
3.2 Regions of optimality in parameter space for minimal DABR coding, $\boldsymbol{p} = (0.58, 0.12, 0.11, 0.1, 0.09)$ 55
3.3 Range of d — ordered by $\frac{1}{1+d}$ — with $[d_{mm}, +\infty]$ shaded 58
3.4 Algebraic minimax redundancy coding, $\boldsymbol{p} = \frac{1}{19} \cdot (8,4,3,2,2)$ (bottom-merge) 64
3.5 Top-merge minimax redundancy coding, $\boldsymbol{p} = \frac{1}{19} \cdot (8,4,3,2,2)$ (single variable) 64

4.1 A simple example of the Package-Merge algorithm 76
4.2 The set of nodes I with widths ($\rho(i,l)$'s) and weights ($\mu(i,l)$'s) for $f(l,p) = pl^2$, $n = 4$, $l_{max} = 3$ 77

4.3 The set of nodes I, an optimal nodeset N, and disjoint subsets A, B, C, D 82

5.1 Sample coding trees for infinite alphabets 91

5.2 More examples of infinite coding trees 94

Chapter 1

Introduction

In source coding problems — often grouped under the term “lossless compression” — the goal is to translate (or “map”) a data object into bits. The method of mapping is termed the *code*, and is often said to consist of a set of *codewords*, each being a string of 0’s and 1’s corresponding to a possible object. The object may be text, image, or other data, but the objective is almost always the smallest data representation possible for storage or transmission. The measure of success is usually the expected number of bits.

1.1 An introductory problem

The game of twenty questions — in which one is challenged to identify an item by asking up to twenty “yes” or “no” questions — is often cited when explaining source coding and entropy, that most basic of information-theoretic concepts [7, 82]. Indeed, the general problem of binary coding, finding a representation of a possibility in terms of sequences of bits — “yes” or “no” questions — is also encountered in twenty questions. By defining different rules for this game, we may present information theoretic topics within the context of the game, yielding formulations for constrained coding and error correction/detection decoding.

However, there is at least one vital difference between standard coding and twenty questions: In the actual game of twenty questions, minimization of mean length is not in fact the goal. The goal instead is to maximize the probability that we name the correct item within twenty questions, that is, to solve

$$\arg\max_{\boldsymbol{l}} P_X\{l(X) \leq 20\} = \arg\max_{\boldsymbol{l}} E_X[1_{l(X)\leq 20}] \tag{1.1}$$

where X is the random variable to be coded, $l(x)$ is the length of the codeword representing instance x, $\boldsymbol{l}$ is the set (or vector) of these lengths, and 1_τ is 1 if τ is true, 0 otherwise. In this form of twenty questions, if there are a very large or infinite number of items to choose from, we should consider only the most likely items. These we should code with all but one of the admissible sequences, leaving the last of the 2^{20} admissible sequences to correspond to "everything else." If p_i is the probability of the ith most likely item, this would yield a success rate of $P_X\{l(X) \leq 20\} = \sum_{i=1}^{2^{20}-1} p_i$.

Twenty questions, although admittedly fabricated, nevertheless reminds us that there are practical situations in which the goal is not achieving the optimal expected rate. Only in coding for high-rate communications is expected rate the precise concern, so we may want to generalize beyond minimizing expected codeword length.

1.2 Motivation

Practical problems in which the goal is not minimizing mean length include those involving remote exploration, military applications, and group testing for diagnostic systems, e.g., blood testing, in which each binary result is like a communicated bit [46, 56]. In such situations, at least one direction of communication may enable only a handful of crucial bits to be sent — a natural channel may have nearly zero bandwidth, a mission-critical channel may at times be jammed by an adversary, and blood tests may be costly and time-consuming. In such situations, it is not clear that mean length should be the measure of success; we find it is often not.

If there are several requests for information, second-order delay effects [58] and finite buffer space [45] may also be considerations. For these queueing-related applications, we assume service time is a linear function of codeword length. In [45], Humblet minimized probability of buffer overflow, considering the reality of limited buffer space. In [58], Larmore considered a case in which the information requests are asynchronous; here long codewords may delay vital communication through a channel due to the "slow truck effect," the phenomenon in which many small packets (i.e., short codewords or fast cars) are held up by one long packet (long codeword or slow truck) [30].

In addition, even a channel that has moderate to high ergodic (Shannon) capacity, may, within certain time periods, not allow a large number of bits to be sent. Users of mobile devices should be familiar with this phenomenon. We may thus put a high priority on receipt of a critical message.

In his 1969 book, *A Diary on Information Theory* [82], Rényi related a similar situation. It is based on an event said to have occurred during the Bar Kochba revolt of AD 132–135, a revolt against Roman plans to build a pagan temple on top of the ruins of the Jewish Temple in Jerusalem. At the time, a siege of indeterminate length restricted the flow of military intelligence to a small number of "yes" and "no" questions. Getting vital information through such questions is not an archaic problem; a recent example of a similar siege with a binary lossless channel was reported in 2002 [70]. We may ask what the coding strategy should be for sending a message necessary to defend a fortress under siege. We will find that this too is a nonlinear problem.

Sequences of bits may be nonlinearly penalized in many instances. To frame such problems, one may use some sort of criterion, or *penalty function*. The penalty is selected according to the value of obtaining the information with a certain number of codeword bits, or, equivalently, to obtaining it with a certain number of tests or in a certain amount of time.

1.3 Formalization

Let us briefly formalize the standard coding problem so that we can properly generalize it to other penalties. Each codeword, or sequence of answers to predetermined questions, is in $\{0,1\}^*$, the set of all finite sequences of 0's and 1's (or "no"s and "yes"s, respectively).

We wish to assign such binary codewords to an alphabet, $\mathcal{X}$. We may assume, without loss of generality, that $\mathcal{X} = \{1, 2, \ldots, n\}$ for some $n = |\mathcal{X}|$ if $|\mathcal{X}| < +\infty$ and $\mathcal{X} = \mathbb{N} \triangleq \{1, 2, \ldots\}$ otherwise. The ith member of $\mathcal{X}$ is thus the integer i, although when we wish to explicitly indicate we are referring to member i, we use m_i.

The random variable we are coding, X, is chosen such that $X = i$ with probability $p_i \in (0, 1]$ for each $i \in \mathcal{X}$. ($\sum_i p_i = 1$.) Probability zero items may be omitted as they never figure in any properties regarding the distribution.

Definition 1 Γ_n^0 *refers to the set of all probability distributions over n items, each of which has strictly positive probability [7]. We may omit the subscript when n is understood.*

Expected values involving functions of X are denoted $E_{\boldsymbol{p}}[f(X)]$ so as to emphasize the underlying probability mass function (or *probability vector*) $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)$. Without loss of generality, we assume $p_i \leq p_j$ if $i > j$. Also, to arrive at a practical algorithm, all p_i's must be assumed to be representable, and thus of finite complexity.

Each item may be assigned a corresponding codeword that we denote as $c_i \in C_{\mathcal{X}} \subset \{0, 1\}^*$, where $C_{\mathcal{X}}$, the code, is the collection of codewords chosen. The only fundamental constraint is that the code be *prefix-free*. That is, to eliminate ambiguity and force codewords to be self-terminating, no codeword may be the prefix of another codeword.

Each codeword c_i is of length l_i, subject to the constraints of the problem. For the sake of notation, we may instead refer to $l(i) = l_i$, viewing it as the ith entry of vector $\boldsymbol{l}$. A code is feasible if and only if its codeword lengths satisfy the Kraft inequality, $\sum_i 2^{-l_i} \leq 1$, as per McMillan's theorem; this applies not just to prefix-free codes but to any uniquely decodable code [21]. Thus all feasible solutions have a prefix-free equivalent. To emphasize connection with the codewords and to avoid confusion with the number 1 or an integer length l, we refer to such a set of feasible codeword lengths as $L_{\mathcal{X}}$ instead of $\boldsymbol{l}$.

Definition 2 *Binary prefix-free codes can be represented by an* encoding binary tree *(or* code tree*), where each path from the root to a leaf represents a distinct code, the*

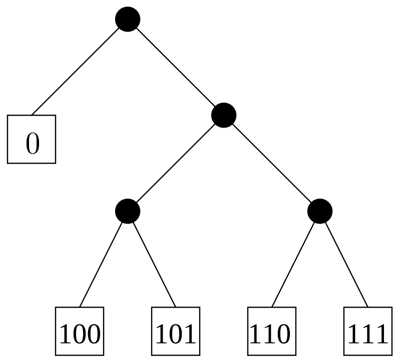


Figure 1.1: A five-codeword prefix-free code $C_{\mathcal{X}}$ represented by a binary tree

kth digit of which is 0 *if the kth edge on the path is the left one and* 1 *if it is the right one. A* full encoding tree *is one that has no nodes with only one child; if a code did not have a full tree a more efficient one could be found by deleting the last bit from a codeword with no sibling. A full tree contains a smaller binary tree — represented in Figure 1.1 by circular nodes or* internal nodes *— and can be viewed as this tree with added leaf nodes, the squares or* external nodes*. In Figure 1.1 we place the resulting codewords in the external node squares to illustrate the relationship between the code and the tree. Such external nodes extend the smaller tree to be full, and such a tree is often called an* extended binary tree*.*

A tree satisfying the Kraft inequality with equality must be full, and a full finite tree satisfies the Kraft inequality with equality. An infinite full tree failing equality is given in [64]; we analyze this tree in Section 5.3.1.

Definition 3 *A collection of lengths, $L_{\mathcal{X}}$, is called a* length ensemble*. (In other papers such collections are referred to as* level sets *[27], a term we avoid due to its different meaning in continuous mathematics, and* leaf patterns *[60], a term we avoid because of its evoking a tree structure, which we do not always desire.)*

We will later show that finding either $C_{\mathcal{X}}$ or $L_{\mathcal{X}}$ solves a coding problem, as a valid code can easily be found from the length ensemble and vice versa. For finite alphabets, when lengths are enumerated in decreasing order and different ensembles for the same n are sorted lexicographically, such sets are well-ordered. (Recall that when we actually refer to lengths by their indices, we sort lengths in increasing order. The ordering of items we use will be made clear from the context.)

Sorted lexicographically, there exists among every set of length ensembles a unique minimum length ensemble. For the set of all length ensembles satisfying the Kraft inequality, this corresponds to the code with $2^{\lceil \lg n \rceil} - n = 2^{\lceil \lg n \rceil} - 2^{\lg n}$ codewords of length $\lfloor \lg n \rfloor$ and $2n - 2^{\lceil \lg n \rceil}$ of length $\lceil \lg n \rceil$. The other extreme is the maximum length ensemble among ensembles satisfying the Kraft inequality with equality, $\{1, 2, \ldots, n-1, n-1\}$. This leads to the following definitions:

Definition 4 *A* minimum length ensemble (minimal code) *is the smallest length ensemble (associated code) among a set when ordered lexicographically. A* flat length ensemble (flat code) *is the smallest length ensemble (code) among all valid ensembles (codes) for an alphabet. A* unary length ensemble (unary code) *is the maximum length ensemble among ensembles (codes) satisfying the Kraft inequality with equality, that is,* $\{1, 2 \ldots, n-1, n-1\}$.

These distinctions become useful when an optimal code is not unique, or when we wish to characterize the solutions for a family of problems.

With the Kraft inequality and perhaps other constraints, Huffman coding [43] finds the codebook $C_{\mathcal{X}} = \{c_i\}$ that minimizes the expected length, $E_{\boldsymbol{p}}[l(X)] = \sum_i p_i l_i$. The central goal of coding theory is this minimization, due to the strong law of large numbers. This law states that the mean length of a sequence of independent and identically distributed occurrences approaches the expected length with probability 1. This property also holds for many sequences that are not identically distributed [25]. Thus, with large amounts of data, if minimization of average length is the goal, one need only worry about expected length, and the penalty is linear.

As noted above, however, we do not always have large amounts of data and average length is not always our goal. The broadest generalization of the problem considered in this dissertation may be stated as the following minimization over length ensembles:

$$\begin{array}{ll} \text{Given} & \boldsymbol{p} \in \Gamma_n^0 \\ & F : \mathbb{N}^n \times [0,1]^n \to \mathbb{R} \cup \{+\infty\} \\ \text{Minimize}_{\{L_{\mathcal{X}}\}} & F(L_{\mathcal{X}}, \boldsymbol{p}) \\ \text{subject to} & \sum_i 2^{-l_i} \leq 1 \\ & l_i \in \mathbb{N}. \end{array} \tag{1.2}$$

Note that if the penalty has a range including $+\infty$, this indicates constraints to the coding problem within the penalty function, as in [12].

For all the cases considered here, we assume that, if $L_{\mathcal{X}} \succeq L'_{\mathcal{X}}$ — that is, $l_i \geq l'_i \; \forall \, i$ — then $F(L_{\mathcal{X}}, \boldsymbol{p}) \geq F(L'_{\mathcal{X}}, \boldsymbol{p})$. All reasonable penalties have this form, and therefore all problems of this form have solutions that satisfy the Kraft inequality with equality. (Note that this does not mean that all minimizing solutions satisfy equality, but that at least one does.)

Although this formulation is too general to be solved as definitively as the linear version, several special cases have been previously solved or otherwise explored. We aim to extend the space of efficiently soluble cases to other problems of special interest, as well as exploring the structure and properties of previously examined cases.

The penalty functions shown in Table 1.1 illustrate the wide range of penalty functions possible. Equation F_1 is the expectation function we are all familiar with, but several others in this table arise in other cases we will explore.

Note that we have assumed we are coding only one piece of information with a known probability mass function, not multiple pieces of information with different probabilities. However, if we assume we have a complete probability model for which events may not be independent or identically distributed, then we may easily extend our results to these cases by considering probability mass function $\boldsymbol{p}$ to take all past events into account. In contrast, if our model is incomplete, nonadaptive universal techniques [21] might apply, but we do not explore these here.

It is noteworthy, too, that adaptive and universal coding techniques are often not the wisest to use in practical instances of coding. In many cases, their small gains over Huffman coding are not worth the trade-off in complexity [11, 93]. For all the aforementioned reasons, here we consider only the problem as given in equation (1.2).

$F_1(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\sum_{i=1}^{n} p_i l_i$	(Huffman case)
$F_2(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\sum_i p_i a^{l_i}$	for some $a > 1$
$F_3(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$-\sum_i p_i a^{l_i}$	for some $0 < a < 1$
$F_4(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\frac{1}{b} \lg \sum_i p_i 2^{b l_i}$	for some $b \neq 0$
$F_5(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\frac{1}{d} \lg \sum_i \left(\frac{p_i^{\frac{1+b+d}{1+b}}}{\sum_j p_j^{\frac{1}{1+b}}} \right) 2^{d l_i}$	for some $b > -1, d \neq 0$
$F_6(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\max_i (l_i + \lg p_i)$	
$F_7(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\prod_i p_i^{-l_i} = 2^{\left(\sum_i l_i \cdot \lg \frac{1}{p_i}\right)}$	
$F_8(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\sum_i p_i^{-l_i}$	
$F_9(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\sum_i p_i (l_i^2 + \gamma l_i)$	for some $\gamma \geq 0$
$F_{10}(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\sum_i p_i l_i^{\alpha}$	for some $\alpha \geq 1$
$F_{11}(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\sum_i (1 - p_i)^{-l_i}$	
$F_{12}(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\sum_i l_i^{(1+p_i)}$	
$F_{13}(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\begin{cases} \sum_i p_i l_i & \text{if } \max_i l_i \leq l_{max} \\ +\infty & \text{otherwise} \end{cases}$	for some $l_{max} \geq \lceil \lg n \rceil$
$F_{14}(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\begin{cases} \frac{\lambda \sum_i p_i l_i^2}{2(1 - \lambda \sum_i p_i l_i)} + \sum_i p_i l_i, & \sum_i p_i l_i < \frac{1}{\lambda} \\ +\infty, & \sum_i p_i l_i \geq \frac{1}{\lambda} \end{cases}$	for some $\lambda > 0$
$F_{15}(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	a^{s^*}, where s^* is the largest s such that $g_T(-s) \cdot \sum_i p_i e^{-s l_i}$	for some $a > 1$, moment generating function g_T
$F_{16}(L_{\mathcal{X}}, \boldsymbol{p})$	$=$	$\prod_i l_i^{p_i} = 2^{\sum_i p_i \lg l_i}$	

Table 1.1: Examples of penalty functions

1.4 Prior work

The literature on aspects of and variants of the Huffman problem is surprisingly vast. For a thorough review, we would refer the reader to surveys on the topic by Abrahams [2, 3]. Here we discuss prior work on alternative penalty functions, on secondary tie-breaking penalty functions, and on infinite alphabets. We ignore cases with further constraints, such as alphabetic codes [42], '1'-ended codes [10, 17], and fix-free codes [98].

The earliest variants of Huffman coding are two deterministic versions of the algorithm, the so-called bottom-merge and top-merge versions. Because codes with different length ensembles may have the same expected length, these linear time algorithms break ties by optimizing any of a number of secondary criteria among codes with minimal average length [27, 38, 53, 55, 68, 84].

The first alternative (primary) penalties were alluded to by Rényi [80] and proposed by Campbell [7, 14, 15]. In this dissertation, we often refer to the Campbell penalties. Campbell's most general formulation was a penalty of the form $F(L_{\mathcal{X}}, \boldsymbol{p}) = f^{-1}\{\sum_i p_i f(l_i)\}$ for some continuous strictly monotonic function $f : \mathbb{R}^+ \to \mathbb{R}^+$. He explored the properties of convex and concave f, and related exponential f — that for which $F(L_{\mathcal{X}}, \boldsymbol{p}) = \log_a \sum_i p_i a^{l_i}$ for positive $a \neq 1$ — to Rényi entropy [81]. A (linear time) Huffman-like algorithm for this penalty was found independently by Hu, Kleitman, and Tamaki [39] for $a > 1$ in the context of search trees, and by Humblet [45] for all $a \neq 1$ to minimize the probability of buffer overflow in a $GI/G/1$ queue (using an asymptotic approximation given by Wyner in [96]). In the latter case, the extension was applied to a more complex penalty, shown as F_{15} in Table 1.1. In practice, Humblet found that the overall algorithm was efficient, though no time complexity bounds were given. Huffman-like coding techniques were also explored in several follow-up papers [18, 53, 78], the first of which also solved a previously proposed problem, the axiomatically derived coding penalty of Nath [76], which, as with Campbell's exponential penalties, calls for the optimization of any one of a singly parameterized set of functions.

Using the linear penalty but limiting maximum length to an integer $l_{max} \geq \lceil \lg n \rceil$

has been thoroughly examined, and may be stated as in F_{13} in Table 1.1. Many papers explore algorithms to find the solution to this problem [31,41,57,92] or approximations [48,72,75]; [72] provides a more comprehensive list of references for this problem. The first efficient solution was given by Larmore and Hirschberg in [59]. This algorithm has $O(nl_{max})$ time and $O(n)$ space. Constant-factor and special-case improvements have been made since [48, 63, 74, 88, 89]. In addition, an asymptotic improvement has been made made by Schieber in [83], a refinement of a previous paper [9] using a different reduction [60]. This $O(n2^{O(\sqrt{\log l_{max} \log\log n})})$ time algorithm has not been put into wide use, however, and no linear time algorithms have been proposed [72].

The approach of Garey in [31] was modified to optimize another practical nonlinear penalty, given as equation F_{14}, that of minimizing delay through an $M/G/1$ queue [58]. The time complexity of this is $O(n^5)$ [space $O(n^3)$], consisting of $O(n^2)$ optimizations of the form $F(L_{\mathcal{X}}, \boldsymbol{p}) = \sum_i p_i(l_i^2 + \gamma l_i)$ for $\gamma \geq 0$ (equation F_9), each with time and space complexity $O(n^3)$; we find we can reduce this complexity.

In [20], as with [39], Huffman trees are viewed as search trees, in this case the goal being to minimize expected maximum search length among multiple concurrently traversed trees. A linear time approximation algorithm is given. Drmota and Szpankowski considered the problem of minimizing maximal pointwise redundancy, $F(L_{\mathcal{X}}, \boldsymbol{p}) = \max_i(l_i + \lg p_i)$, in [24], offering a $\Theta(n \log n)$ solution generalizing Shannon coding [85].

Note that those problems for which polynomial time solutions have been offered are specific problems or problems parameterized by one real variable. (Humblet's buffer overflow problem, although efficient in practice, has not been shown to be polynomial time in the worst case, and it ultimately reduces to the singly parameterized exponential case.) We consider and solve some more general classes here.

Huffman coding for infinite alphabets has been primarily examined for cases in which probabilities of items decline precisely geometrically, and thus can be optimally encoded by the composition of a finite code and a unary code [1, 19, 28, 35, 36, 71]. The only other method of producing such codes works only for cases in which the rate of fall-off of the probabilities is rapidly exponential or superexponential [44, 49]. Linder, Tarokh, and Zeger proved in [64] that optimal codes always exist, but this

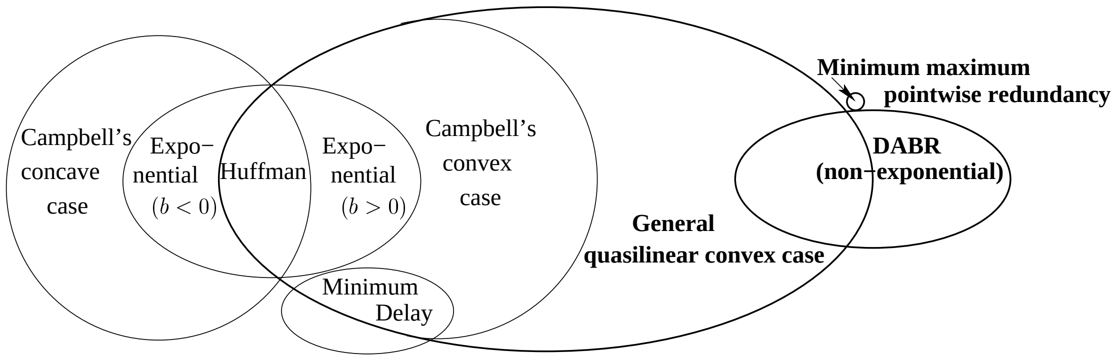


Figure 1.2: Penalty classes we consider in this dissertation

is nonconstructive. No previously published work has explored alternate penalties in the infinite case.

1.5 Organization

Figure 1.2 illustrates many penalty classes we consider in this dissertation, with ones that we both solve and examine in significant depth in bold. Properties of the other penalties shown are also explored, as is the infinite alphabet case.

In Chapter 2 we discuss a nonlinear penalty that is an extension of standard coding — the exponential penalty — one which introduces a parametric degree of freedom. Its algorithmic solution and properties are discussed. In Chapter 3 we extend this penalty to consider a measurement of deviation of actual codeword length from ideal; this adds a second degree of freedom, framing previously known problems in a larger context, that of d-average b-redundancy (or DABR). In Chapter 4 we consider a further generalization that solves an entire family of problems, the quasilinear convex family. Finally, in Chapter 5 we extend the aforementioned results to infinite alphabets representing a countable number of outcomes.

Chapter 2

Exponential penalties

In this chapter, we introduce exponential penalties and investigate their algorithmic solutions, properties, and applications.

2.1 Nonlinear penalties

With a nonlinear penalty, one for which minimization of expected length is not the goal, the solution for a coding problem may differ from that of the linear case. In certain instances, choosing the penalty may be more a matter of art than science, although many nonlinear problem formulations have precise specifications. The linear penalty, that of expectation, is the one that yields the results concerning most of coding theory. We might expect this to be one end of a spectrum of possibilities, away from which we are penalized more and more extremely for long codewords. As before, penalty should be a function of lengths of the codewords, but not necessarily a linear one.

We previously stated that finding either $C_{\mathcal{X}}$ or $L_{\mathcal{X}}$ solves the problem. Obviously, $L_{\mathcal{X}}$ may be obtained from $C_{\mathcal{X}}$. Given $L_{\mathcal{X}}$ — the collection of lengths $l_i \in \mathbb{N}$ — satisfying the Kraft inequality (the only explicit constraint here), we can construct codewords for these lengths by building a tree in top-down fashion to match lengths enumerated in decreasing order. We briefly describe how.

Consider a well-known unpublished coding technique of Elias [33]. View each

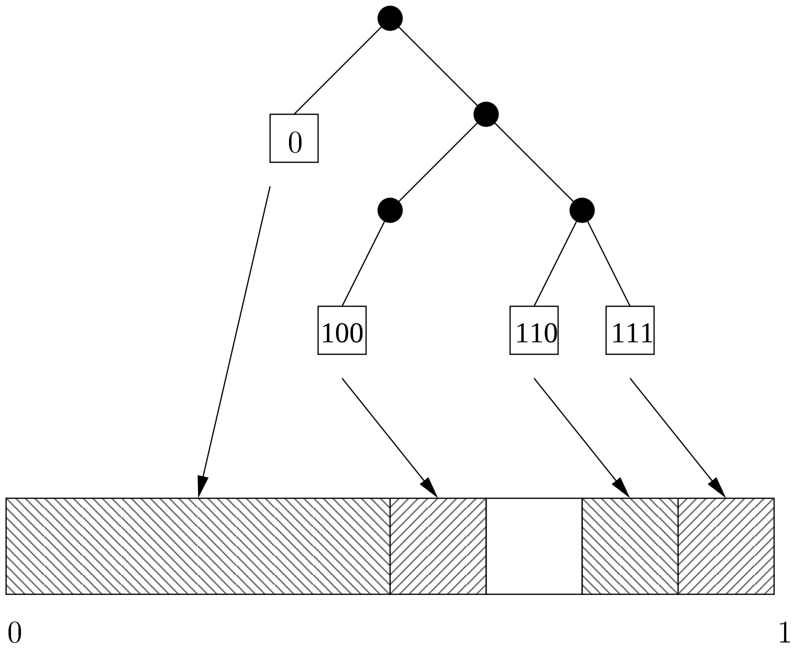


Figure 2.1: Representations of a prefix-free code as a tree and as subintervals of $[0, 1)$

codeword c_i of length l_i as occupying a portion of the $[0, 1)$ interval beginning at its binary fractional expansion and occupying a subinterval of length 2^{-l_i}, as in Figure 2.1. Thus, for example, the codeword 101 would occupy $[0.625, 0.75)$. It is clear that these intervals are disjoint for codewords of a prefix-free code and that their summed length is the Kraft inequality sum. With l_i sorted in increasing order, then, we may let each length define its corresponding codeword by occupying the next portion of the unit interval in this order. This works even if $\mathcal{X}$ is infinite.

Thus we only need to find the values of l_i that optimize the criteria to find an optimal code. The goal should be to minimize a function of $L_{\mathcal{X}}$ and $\boldsymbol{p}$. The most general case, as in equation (1.2), is to consider an arbitrary $F(L_{\mathcal{X}}, \boldsymbol{p})$. Slightly more specific is the set of functions of the form $F(L_{\mathcal{X}}, \boldsymbol{p}) = \sum_i f(l_i, p_i)$ for some function $f : \mathbb{N} \times [0, 1] \to \mathbb{R} \cup \{+\infty\}$.

Within this formulation, we always assume f is a monotonically increasing function of l_i. In addition, we usually assume it to be convex, so that there is a penalty for each additional bit and the penalty for each additional bit is at least as large as that of the previous bit. (Convexity refers to increasing derivatives or differences, so that, for $x \in \mathbb{N}$, $g : \mathbb{N} \to \mathbb{R} \cup \{+\infty\}$ is convex if and only if, $\forall\, x \in \mathbb{N}, g(x+1) \leq \frac{g(x+2)+g(x)}{2}$.)

The convexity assumption is sensible for many examples in which the bits sent are critical. Additional bits are penalized more heavily in all previously explored applications, although we will propose a case where this is not so.

Given such an f and a $\boldsymbol{p}$, the problem is:

$$\begin{array}{lll} \text{Minimize}_{\{L_{\mathcal{X}}\}} & \sum_i f(l_i, p_i) & \\ \text{subject to} & \sum_i 2^{-l_i} \leq 1 & \text{(Kraft inequality)} \\ & l_i \in \mathbb{N} & \text{(integer constraint)} \end{array} \tag{2.1}$$

We call penalties of this form *generalized quasilinear* penalties.

A class proposed by Campbell in [15] (and previously hinted at by Rényi in [80]) covers most interesting penalties of this form. He considered F's of the form $F(L_{\mathcal{X}}, \boldsymbol{p}) = \sum_i p_i f(l_i)$, or $E_{\boldsymbol{p}}[f(l(X))]$. We may also write this as $F(C_{\mathcal{X}}, \boldsymbol{p})$, given code $C_{\mathcal{X}}$. The Campbell class is a *quasilinear mean value formulation* [8]. (The mean is also known as a weighted quasiarithmetic mean [7], a Kolmogorov-Nagumo mean [80], or a de Finetti-Kitagawa mean [4]). The problem may be stated as follows:

$$\begin{array}{ll} \text{Given} & \boldsymbol{p} \in \Gamma^0 \\ & \text{strictly monotonically increasing } f(x) \\ \text{Minimize}_{\{L_{\mathcal{X}}\}} & E_{\boldsymbol{p}}[f(l(X))] \\ \text{subject to} & \sum_i 2^{-l_i} \leq 1 \\ & l_i \in \mathbb{N} \end{array} \tag{2.2}$$

The proper quasilinear formulation is

$$\begin{array}{ll} \text{Given} & \boldsymbol{p} \in \Gamma^0 \\ & \text{strictly monotonic } \tilde{f} \in \mathcal{C}^0 : \mathbb{R}^+ \to \mathbb{R}^+ \\ \text{Minimize}_{\{L_{\mathcal{X}}\}} & \tilde{f}^{-1}\{E_{\boldsymbol{p}}[\tilde{f}(l(X))]\} \\ \text{subject to} & \sum_i 2^{-l_i} \leq 1 \\ & l_i \in \mathbb{N} \end{array} \tag{2.3}$$

which is more useful for analysis — as we will see in Sections 4.2 and 5.3.1 — although it does not cover cases where any values are $+\infty$. It does, however, allow $\tilde{f}$ to be monotonically decreasing. Note that, because $f^{-1}\{E_{\boldsymbol{p}}[\tilde{f}(l(X))]\} \geq 1$, we need consider only the domain $[1, +\infty]$, so, if $\tilde{f}$ is monotonically decreasing, we may use the form of (2.2) with $f(x) = f(1) - \tilde{f}(x)$. Thus, our initial assumption that f is monotonically increasing holds without loss of generality. For continuous monotonic $f : \mathbb{R}^+ \to \mathbb{R}^+$, statements (2.2) and (2.3) may be used interchangeably, as the case warrants.

Throughout this dissertation we assume there is no constraint on the codeword values not implicit in the penalty function. Note that any increasing transformation of $F(L_{\mathcal{X}}, \boldsymbol{p})$ leaves the solution unchanged, as the minimizing argument for the objective function $F(L_{\mathcal{X}}, \boldsymbol{p}) = E_{\boldsymbol{p}}[f(l(X))]$ remains the same. Thus, in the linear case, although the function is $f(x) = x$, any $f(x) = \alpha x + \beta$ ($\alpha > 0$, $\beta \in \mathbb{R}$) would also do. For each additional bit used, we exact a constant penalty. This, for finite cases, has the well-known constructive form due to Huffman [43]. Note that, if $2^{-L_{\mathcal{X}}}$ denotes the probability vector for which the ith probability is 2^{-l_i}, the linear problem is equivalent to $\arg\min_{L_{\mathcal{X}}} D(\boldsymbol{p} \parallel 2^{-L_{\mathcal{X}}})$ [65].

2.2 An exponential penalty

2.2.1 Motivation

Since only the linear case has a constant derivative, in all other cases penalty per bit must be a function of length. One possible penalty is $f(x) = 2^{bx} = a^x$ for some parameter $b = \lg a \in (0, +\infty)$. For now we only consider values in the this range because $b \in (-\infty, 0]$ would cause $f(x)$ to be nonincreasing and thus the problem to be ill-posed if put in form (2.2). We choose the parametric variable to be either a or b as appropriate to context. We may thus refer to the objective function as $F_a(L_{\mathcal{X}}, \boldsymbol{p}) = E_{\boldsymbol{p}}[a^{l(X)}]$ or $F_b(L_{\mathcal{X}}, \boldsymbol{p}) = E_{\boldsymbol{p}}[2^{bl(X)}]$. We also find it useful at times to map the parameter to the unit interval, assigning

$$\alpha \triangleq \frac{1}{1+b} = \frac{1}{1+\lg a} \in (0,1). \tag{2.4}$$

We may also drop the parameter subscript for $F(L_{\mathcal{X}}, \boldsymbol{p})$ if the implicit values are clear from context.

Instead of a constant additional penalty, the penalty is now a geometric one. For example, one additional bit would exact a 10% penalty over the shorter code for the case of $a = 1.1$. If we want to place a high penalty on additional bits, we choose a large parameter, whereas if we want to place a penalty that increases only slightly per bit with additional bits, we choose a small parameter.

Let us define

$$G_b(L_{\mathcal{X}}, \boldsymbol{p}) \triangleq \frac{1}{b} \lg E_{\boldsymbol{p}}[2^{bl(X)}] \tag{2.5}$$

so that we may put this problem in the quasilinear form given in equation (2.3), that is, as $\arg\min_{L_{\mathcal{X}}} G(L_{\mathcal{X}}, \boldsymbol{p})$. The formulation using G is often called a β-exponential average or β-average for $b = \beta$ [7]. Note that we may expand the domain to $[0, +\infty]$ by instead defining

$$G_b(L_{\mathcal{X}}, \boldsymbol{p}) \triangleq \lim_{\beta \to b} \frac{1}{\beta} \lg E_{\boldsymbol{p}}[2^{\beta l(X)}]. \tag{2.6}$$

Then

$$\begin{array}{rclcl} G_0(L_{\mathcal{X}}, \boldsymbol{p}) & = & \lim_{\beta \to 0} \left[E_{\boldsymbol{p}}(l(X)) + \frac{\beta}{2}\sigma^2_{\boldsymbol{p}}(l(X)) + O(\beta^2)\right] & = & E_{\boldsymbol{p}}(l(X)) \\ G_{+\infty}(L_{\mathcal{X}}, \boldsymbol{p}) & = & \lim_{\beta \uparrow +\infty} \left[\max_i l_i + \frac{1}{\beta} \lg P_{max} + O(\frac{1}{\beta 2^\beta})\right] & = & \max_i l_i \end{array} \tag{2.7}$$

where $P_{max} = P_X\{l(X) = \max_j l_j\}$, $\sigma^2_{\boldsymbol{p}}(X) = E_{\boldsymbol{p}}\{[X - E_{\boldsymbol{p}}(X)]^2\}$, and $O(u(\beta))$ denotes a $v(\beta)$ such that $\lim_{\beta \to b} \frac{|v(\beta)|}{|u(\beta)|}$ exists and is finite. (Constant b, usually $+\infty$ though often $-\infty$ or 0, is always implied by the context. For example, it is $+\infty$ when discussing algorithm complexity. Also, $\Theta(u(\beta))$ may be used instead if the limit is additionally nonzero.)

In other words, as $b \uparrow +\infty$ ($a \uparrow +\infty$), minimizing objective function $F_{+\infty}(L_{\mathcal{X}}, \boldsymbol{p})$ becomes minimizing the maximum length, and, given all possible codes minimizing maximum length, this chooses the one that minimizes the probability that this maximum length is achieved. This corresponds to coding using the flat length ensemble. Similarly, as $b \downarrow 0$ ($a \downarrow 1$), $\arg\min_{L_{\mathcal{X}}} E_{\boldsymbol{p}}[2^{bl(X)}] \to \arg\min_{L_{\mathcal{X}}} E_{\boldsymbol{p}}[l(X)]$, the Huffman case, and, given multiple possible Huffman lengths, this chooses the one with the minimum variance, $\sigma_{\boldsymbol{p}}^2$; we return to this later. Thus we have defined a problem with a parameter that varies solution values between these two extreme cases, trading off between minimum average length and minimum maximum (minimax) length.

Penalties $F_b(L_{\mathcal{X}}, \boldsymbol{p})$ and $G_b(L_{\mathcal{X}}, \boldsymbol{p})$ are nondecreasing in b (and thus a). The nondecreasing property of G may be seen by noting that, for all $\boldsymbol{v} = \{v(i)\}$, the moment about zero $M_{\boldsymbol{p}}^{v}(d) = \{E_{\boldsymbol{p}}[v(X)]^d\}^{1/d}$ is nondecreasing in d. Since $G_b(L_{\mathcal{X}}, \boldsymbol{p}) = \lg M_{\boldsymbol{p}}^{2^{L_{\mathcal{X}}}}(b)$, this is nondecreasing in b. This implies the same for F.

In addition to the above properties and the simplicity of the exponential penalty, we find that, compared to other reasonable alternatives, such as $f(x) = x^a$, the penalty $f(x) = a^x$ guards against unusually long codewords to a far greater degree. Another reasonable alternative, limiting the maximum length and optimizing for expected length within this constraint, was considered in [59], which found an efficient algorithm for this problem. We return to such a case later, but for now we dismiss it for not having the smoothness of trade-off we desire.

The exponential is the traditional penalty of risk sensitivity (risk aversion) [94] and satisfies a number of mathematical properties in this context [7], such as additivity and translativity, to be discussed. In addition, this penalty may be used in minimizing probability of buffer overflow for variable length code transmission [45]. Further justification for this penalty has been given by Aczél and Dhombres in [8]. Thus, we first consider $f(x) = a^x$, and only later consider other penalties.

2.2.2 Real-valued problem solution and bounds

The exponential penalty was introduced by Campbell [14, 15] and algorithmically solved in [39,45]. Here we present a more detailed examination to the problem, based on the approach towards the linear penalty given by Cover and Thomas in [21]. We also look at properties of and extensions to the exponential problem.

To get an idea as to the form for the minimization over $L_{\mathcal{X}}$, we should first consider the problem where we do not restrict the values of l_i to integers; call this the optimal *ideal length* sequence, $L_{\mathcal{X}}^{\dagger} = \{l_i^{\dagger}\}$. In this case, the problem is one of constrained minimization, which we may rewrite using Lagrange multipliers as the minimization of:

$$J \triangleq \sum_i p_i a^{l_i} + \lambda \left(\sum_i 2^{-l_i} \right) = \sum_i p_i 2^{b l_i} + \lambda \left(\sum_i 2^{-l_i} \right). \tag{2.8}$$

Differentiating with respect to l_i, we obtain

$$\frac{\partial J}{\partial l_i} = b p_i 2^{b l_i} \ln 2 - \lambda 2^{-l_i} \ln 2. \tag{2.9}$$

If we set the derivative to 0 and solve for l_i, we find

$$2^{-l_i^\dagger} = \left(\frac{b p_i}{\lambda} \right)^{\frac{1}{1+b}} \tag{2.10}$$

or

$$l_i^\dagger = \lg \left[\left(\frac{b p_i}{\lambda} \right)^{-\frac{1}{1+b}} \right] = \frac{-\lg \frac{b p_i}{\lambda}}{1+b} = -\frac{1}{1+b} \lg b p_i + \frac{1}{1+b} \lg \lambda \tag{2.11}$$

where λ is the solution of

$$\sum_j \left(\frac{b p_j}{\lambda} \right)^{\frac{1}{1+b}} = \lambda^{-\frac{1}{1+b}} b^{\frac{1}{1+b}} \sum_j p_j{}^{\frac{1}{1+b}} = 1 \tag{2.12}$$

which is

$$\lambda = b \cdot \left(\sum_j p_j{}^{\frac{1}{1+b}} \right)^{1+b} \tag{2.13}$$

yielding

$$l_i^\dagger = -\frac{1}{1+b} \lg b p_i + \frac{1}{1+b} \lg \left[b \cdot \left(\sum_j p_j{}^{\frac{1}{1+b}} \right)^{1+b} \right] = -\frac{1}{1+b} \lg p_i + \lg \sum_j p_j{}^{\frac{1}{1+b}} \tag{2.14}$$

or

$$2^{-l_i^\dagger} = \frac{p_i^{\frac{1}{1+b}}}{\sum_j p_j{}^{\frac{1}{1+b}}} = \frac{p_i^{\frac{1}{1+\lg a}}}{\sum_j p_j{}^{\frac{1}{1+\lg a}}}. \tag{2.15}$$

The value for the objective function at this extremum is

$$F(L^\dagger_{\mathcal{X}}, \boldsymbol{p}) = E_{\boldsymbol{p}}[2^{bl^\dagger(X)}] = E_{\boldsymbol{p}}\left[\left(\frac{p(X)^{\frac{1}{1+b}}}{\sum_j p(j)^{\frac{1}{1+b}}}\right)^{-b}\right] = \left(\sum_j p_j{}^{\frac{1}{1+b}}\right)^{1+b} = 2^{bH_\alpha(\boldsymbol{p})} \tag{2.16}$$

or

$$G(L^\dagger_{\mathcal{X}}, \boldsymbol{p}) = \frac{1}{b} \lg F(L^\dagger_{\mathcal{X}}, \boldsymbol{p}) = H_\alpha(\boldsymbol{p}) \tag{2.17}$$

where we recall $\alpha = \frac{1}{1+b} = \frac{1}{1+\lg a}$ and

$$H_\alpha(\boldsymbol{p}) \triangleq \frac{1}{1-\alpha} \lg \sum_i p_i^\alpha, \tag{2.18}$$

the Rényi entropy of order α for this distribution. This should not be surprising given the relationship between Huffman coding and Shannon entropy, which corresponds to $b \to 0$ and $H_1(\boldsymbol{p})$ [85].

This problem is a convex optimization problem, and one can easily confirm that it satisfies the Karush-Kuhn-Tucker optimality conditions. Thus this local extremum is the global minimum [12].

Thus, $2^{-l_i^\dagger}$ for each codeword should be proportional to a power of the corresponding p_i, this power being a function of the constraint parameter. Note that because probabilities sum to 1, we can invert equation (2.14) to obtain the probability mass function ideally corresponding to a given $L_{\mathcal{X}}$:

$$p_i = \frac{2^{-(1+b)l_i}}{\sum_j 2^{-(1+b)l_j}} = \left[\sum_j (2a)^{(l_i - l_j)}\right]^{-1} \tag{2.19}$$

2.2.3 Integer-valued problem solution and algorithm

Due to the integer constraint, the optimal codeword lengths $L^*_{\mathcal{X}}$ may not be the ideal $L^\dagger_{\mathcal{X}}$. A suboptimal solution is $l_i^S = \lceil l_i^\dagger \rceil$, which satisfies the Kraft inequality, and may be specified for both finite and infinite alphabets. This is similar to Shannon coding and we thus call it a *Shannon-like code*. For the linear penalty, Shannon coding is $l_i^S = \lceil -\lg p_i \rceil$, which is optimal iff the problem is dyadic, that is, all $p_i \in 2^{\mathbb{Z}}$, the set of powers of two. Similarly, for this more general case, an optimal solution is obtained — and equality thus achieved — iff all $l^\dagger$'s specified by (2.14) are integers.

Thus we have

$$G(L_{\mathcal{X}}^\dagger, \boldsymbol{p}) \leq G(L_{\mathcal{X}}^*, \boldsymbol{p}) \leq G(L_{\mathcal{X}}^S, \boldsymbol{p}) < 1 + H_\alpha(\boldsymbol{p}). \tag{2.20}$$

since $G(L_{\mathcal{X}}^S, \boldsymbol{p}) = \frac{1}{b} \lg E_{\boldsymbol{p}}[2^{b\lceil l^\dagger(X)\rceil}]$ and $1 + H_\alpha(\boldsymbol{p}) = \frac{1}{b} \lg E_{\boldsymbol{p}}[2^{b[1+l^\dagger(X)]}]$. This leads to

$$\begin{aligned} H_\alpha(\boldsymbol{p}) &\leq G(L_{\mathcal{X}}^*, \boldsymbol{p}) < H_\alpha(\boldsymbol{p}) + 1 \quad (2.21)\\ a^{H_\alpha(\boldsymbol{p})} &\leq F(L_{\mathcal{X}}^*, \boldsymbol{p}) < a^{H_\alpha(\boldsymbol{p})+1}. \quad (2.22)\end{aligned}$$

These inequalities provide the coding-motivated definition of Rényi entropy given by Campbell in [15]. Note that, because ideal optimal lengths depend on all probabilities, it would be difficult to improve on this bound to obtain a bound by knowing only one or two item probabilities, in the manner of [29, 98], among others.

To solve the problem of finding the optimal code for finite alphabets, we need to find a method for formulating a constructive code similar to that devised by Huffman. We are not guaranteed that an efficient algorithm exists given a convex optimization problem restricted to the integers. Finding the optimal solution of a convex optimization problem with integer constraints is a problem with $\mathcal{NP}$-hard special cases, such as the nearest lattice vector problem [37, 90], integer programming [32], and the knapsack problem [32,66]. In addition, for coding, the size of the solution space grows exponentially with the number of items to be considered [73, 77]. However, we can take advantage of the special structure of this problem in order to find an efficient solution.

Various approaches have been used to understand the solution to this problem [18, 53, 55, 78, 87], first presented in [39, 45]. We present a fresh approach and look at the consequences of the solution.

The following lemma details some of the structure of the problem. For reasons that will later become clear, we waive the requirement that probabilities sum to 1.

Freed from this restraint, we can address a slightly wider class of problems, scaling the elements to sum to 1 if desired.

Definition 5 *A probability vector that need not have total measure* 1*, but which still has all elements nonnegative, is referred to as a vector of* weights *or* weight vector *and is denoted by* $\boldsymbol{w}$ *instead of* $\boldsymbol{p}$.

Lemma 1 *For every monotonically increasing penalty,* $F(L_{\mathcal{X}}, \boldsymbol{w}) = \sum_i w_i f(l_i)$*, there exists an optimal code such that:*

1. *If* $w_j \geq w_k$*, then* $l_j \leq l_k$.
2. *The two longest codewords have the same length.*
3. *The two longest codewords differ only in the last bit and correspond to the two symbols with the lowest weight.*

The proof is a modified version of that presented for the linear case in [21] as Lemma 5.8.1, which considers codes as binary tree structures.

Proof: Consider an optimal code $C_{\mathcal{X}}$:

1. *If* $w_j \geq w_k$*, then* $l_j \leq l_k$. Given $w_j > w_k$, consider $C'_{\mathcal{X}}$ with the codewords j and k of $C_{\mathcal{X}}$ interchanged. Then

$$\begin{aligned} F(C'_{\mathcal{X}}, \boldsymbol{w}) - F(C_{\mathcal{X}}, \boldsymbol{w}) &= \sum_i w_i f(l'_i) - \sum_i w_i f(l_i) && (2.23)\\ &= w_j f(l_k) + w_k f(l_j) - w_j f(l_j) - w_k f(l_k) && (2.24)\\ &= (w_j - w_k)(f(l_k) - f(l_j)). && (2.25) \end{aligned}$$

But $w_j - w_k > 0$, and since $C_{\mathcal{X}}$ is optimal, $F(C'_{\mathcal{X}}, \boldsymbol{w}) - F(C_{\mathcal{X}}, \boldsymbol{w}) \geq 0$, so $f(l_k) \geq f(l_j)$ and thus $l_k \geq l_j$. Thus $C_{\mathcal{X}}$ itself satisfies $w_j > w_k \Rightarrow l_j \leq l_k$. Furthermore, without loss of generality, we may reorder any identical values of l_i such that, for $w_j \geq w_k$, $l_j \leq l_k$ (and thus, for all $i > j$, $l_i \geq l_j$).

2. *The two longest codewords are of the same length.* If the two longest codewords are not of the same length, then we can delete the last bit of the longer one, preserving the prefix property of prefix-free codes and achieving a strictly lower value for $F(L_{\mathcal{X}}, \boldsymbol{w})$.

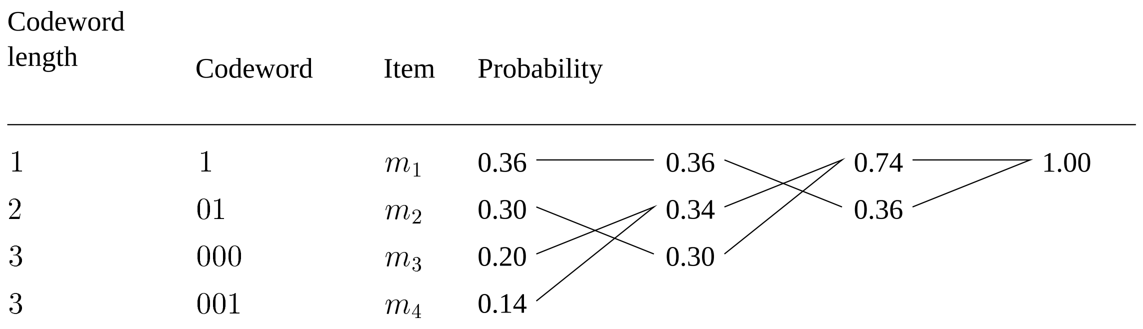
3. *The two longest codewords differ only in the last bit and correspond to the two symbols with the lowest weight.* As shown in the proof of property 1, the longest codewords must belong to the source symbols of least weight. If there is a maximal length codeword without a sibling, then we can delete the last bit of the codeword length and still satisfy the prefix property, as in the proof of property 2. This would reduce $F(L_{\mathcal{X}}, \boldsymbol{w})$ and contradicts optimality of the code. Thus *every* maximal length codeword in any optimal code has a sibling.

   Now we can exchange the longest length codewords so the two lowest probability source symbols are associated with two siblings on the tree. This preserves $L_{\mathcal{X}}$ and thus $F(L_{\mathcal{X}}, \boldsymbol{w})$. Thus the codewords for the two lowest probability source symbols have maximal length and agree in all but the last bit.

■

With Lemma 1, we can, as in [21], restrict the search to codes satisfying these properties. Note that we have eliminated some possibly optimal codes $C_{\mathcal{X}}$ (and thus their equivalent trees) from consideration. However, these codes are, in some sense, equivalent to ones we do consider; we only switch subtrees (prefixes) of equal depth (properties 1 and 3), so, ignoring order, all optimal $L_{\mathcal{X}}$ are still under consideration.

We know an optimal order for a given set of lengths is nondecreasing. We thus have restricted ourselves to the set of sequences of $|\mathcal{X}|$ lengths in this order satisfying the Kraft inequality with equality. Call this set $\mathcal{L}_{|\mathcal{X}|}$, the set of compact codes. Although this set grows quickly with the size of $\mathcal{X}$ [73, 77], we may construct an iterative algorithm from the restrictions of Lemma 1. We term this algorithm *exponential Huffman coding*, which we illustrate by comparison to Huffman coding, algorithmically below and by example in Figures 2.2 and 2.3. (Because this is a generalization of Huffman coding, we may call omit the qualifier "exponential" if it is clear from context.)



Codeword length	Codeword	Item	Probability
1	1	m_1	0.36
2	01	m_2	0.30
3	000	m_3	0.20
3	001	m_4	0.14

Figure 2.2: Huffman coding, $\boldsymbol{p} = (0.36, 0.30, 0.20, 0.14)$

Huffman coding, we recall, has the following constructive algorithm for minimizing expected length:

Procedure for Huffman Coding

1. Each item $m_i \in \{m_1, m_2, \ldots, m_n\}$ has weight $w_i \in \mathcal{W}_{\mathcal{X}}$, where $\mathcal{W}_{\mathcal{X}}$ is the set of all such weights. (Initially, $m_i = i$.) Assume each item m_i has codeword c_i, to be determined later.

2. Combine the items with the two smallest weights w_j and w_k into one item $\tilde{m}_j$ with the combined weight $\tilde{w}_j = w_j + w_k$. This item has codeword $\tilde{c}_j$, to be determined later, while m_j is assigned code $c_j = \tilde{c}_j 0$ and m_k code $c_k = \tilde{c}_j 1$. Since these have been assigned in terms of $\tilde{c}_j$, replace w_j and w_k with $\tilde{w}_j$ in $\mathcal{W}$ to form $\mathcal{W}_{\tilde{\mathcal{X}}}$. $\sum_{\mathcal{W}} w = \sum_{i \in \tilde{\mathcal{X}}} \tilde{w}_i$ remains constant.

3. Repeat procedure, now with the remaining $n-1$ codewords and corresponding weights in $\mathcal{W}$, until only one item is left. The weight of this item is $\sum_{i \in \mathcal{X}} w_i$. All codewords are now defined by assigning the null string to this trivial item.

An example of Huffman coding is shown in Figure 2.2. Exponential Huffman coding is similar, with minor alterations:

Codeword length	Codeword	Item	Weight
2	00	m_1	0.36
2	01	m_2	0.30
2	10	m_3	0.20
2	11	m_4	0.14

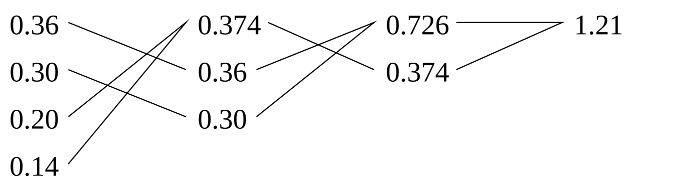


Figure 2.3: Exponential Huffman coding, $\boldsymbol{p} = (0.36, 0.30, 0.20, 0.14)$ and $a = 1.1$

Procedure for Exponential Huffman Coding (Parameter $a = 2^b$)

1. Each item $m_i \in \{m_1, m_2, \ldots, m_n\}$ has weight $w_i \in \mathcal{W}_{\mathcal{X}}$, where $\mathcal{W}_{\mathcal{X}}$ is the set of all such weights. (Initially, $m_i = i$.) Assume each item m_i has codeword c_i, to be determined later.

2. Combine the items with the two smallest weights w_j and w_k into one item $\tilde{m}_j$ with the combined weight $\tilde{w}_j = \phi(w_j, w_k) = a \cdot (w_j + w_k)$ (for ϕ defined in this manner). This item has codeword $\tilde{c}_j$, to be determined later, while m_j is assigned code $c_j = \tilde{c}_j0$ and m_k code $c_k = \tilde{c}_j1$. Since these have been assigned in terms of $\tilde{c}_j$, replace w_j and w_k with $\tilde{w}_j$ in $\mathcal{W}$ to form $\mathcal{W}_{\tilde{\mathcal{X}}}$. $\sum_{\mathcal{W}} w = \sum_{i \in \tilde{\mathcal{X}}} \tilde{w}_i$ need not remain constant.

3. Repeat procedure, now with the remaining $n - 1$ codewords and corresponding weights in $\mathcal{W}$, until only one item is left. The weight of this item is $\sum_{i \in \mathcal{X}} w_i a^{l_i}$. All codewords are now defined by assigning the null string to this trivial item.

We later show that this algorithm may be modified to run in linear time (to input size) given sorted weights in the same manner as Huffman coding [91]. An example of exponential Huffman coding for $a = 1.1$ is shown in Figure 2.3. The resulting code is different from that in Figure 2.2 due to the different penalty. Also note that $F(L_{\mathcal{X}}, \boldsymbol{p}) = E_{\boldsymbol{p}}[1.1^{l(X)}] = E_{\boldsymbol{p}}[1.1^2] = E_{\boldsymbol{p}}[1.21] = 1.21$, the value minimized, is the value calculated for the final combined weight. In the case of $a = 1$, the value is 1 and the method is Huffman coding, as we might expect from (2.7).

Note that possible ties in weight mean that this algorithm, as with Huffman coding, is nondeterministic. Not only may a tie render an optimal solution nonunique in terms of the lengths of its codewords, but, since different codes may correspond to the same length ensembles, the codes themselves may be of many different equivalent forms. As mentioned previously and illustrated later, some optimal codes cannot be derived from (exponential) Huffman coding.

Before we concern ourselves with this, however, we should first prove that the algorithm's result is optimal. A proof of optimality follows for $a > 1$, a proof which follows the format of the proof of Huffman coding optimality in [21], but has a slightly different approach.

Theorem 1 *The exponential Huffman algorithm is optimal, i.e., if C^* is an exponential Huffman code for weights (probabilities) $\boldsymbol{w}$ and parameter $a > 1$, and, if C is any other uniquely decodable code, then $F_a(C^*, \boldsymbol{w}) \leq F_a(C, \boldsymbol{w})$.*

Proof: Assume $C_{\mathcal{X}}$ satisfies the properties of Lemma 1. Define a "merged" code for $n-1$ symbols as follows: Take the common prefix of the two longest codewords (corresponding to the two least values of w_i) and allot it to a symbol with weight $\tilde{w}_{n-1} = a(w_{n-1} + w_n)$. All other codewords remain the same. Consider this new alphabet $\tilde{\mathcal{X}}$ with weight vector $\tilde{\boldsymbol{w}}$. The correspondence is shown below:

$\mathcal{W}_{\tilde{\mathcal{X}}}$	$C_{\tilde{\mathcal{X}}}$	$L_{\tilde{\mathcal{X}}}$	$C_{\mathcal{X}}$	$L_{\mathcal{X}}$
w_1	$\tilde{c}_1$	$\tilde{l}_1$	$c_1 = \tilde{c}_1$	$l_1 = \tilde{l}_1$
w_2	$\tilde{c}_2$	$\tilde{l}_2$	$c_2 = \tilde{c}_2$	$l_2 = \tilde{l}_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_{n-2}	$\tilde{c}_{n-2}$	$\tilde{l}_{n-2}$	$c_{n-2} = \tilde{c}_{n-2}$	$l_{n-2} = \tilde{l}_{n-2}$
$\tilde{w}_{n-1} = (w_{n-1} + w_n)a$	$\tilde{c}_{n-1}$	$\tilde{l}_{n-1}$	$c_{n-1} = \tilde{c}_{n-1}0$	$l_{n-1} = \tilde{l}_{n-1} + 1$
			$c_n = \tilde{c}_{n-1}1$	$l_n = \tilde{l}_{n-1} + 1$

where c_i denotes a binary codeword and l_i its length. The expected penalty of the code $C_{\mathcal{X}}$ with weights $\mathcal{W}_{\mathcal{X}}$ is

$$\begin{aligned}
F(C_{\mathcal{X}}, \boldsymbol{w}) &= \sum_{i=1}^{n} w_i a^{l_i} && (2.26)\\
&= \sum_{i=1}^{n-2} \tilde{w}_i a^{\tilde{l}_i} + w_{n-1} a^{\tilde{l}_{n-1}+1} + w_n a^{\tilde{l}_{n-1}+1} && (2.27)\\
&= \sum_{i=1}^{n-2} \tilde{w}_i a^{\tilde{l}_i} + (w_{n-1} + w_n) a \cdot a^{\tilde{l}_{n-1}} && (2.28)\\
&= \sum_{i=1}^{n-1} \tilde{w}_i a^{\tilde{l}_i} && (2.29)\\
&= F(C_{\tilde{\mathcal{X}}}, \tilde{\boldsymbol{w}}). && (2.30)
\end{aligned}$$

Thus, given the conclusions of Lemma 1, we have reduced minimizing $F(C_{\mathcal{X}}, \boldsymbol{w})$ to minimizing $F(C_{\tilde{\mathcal{X}}}, \tilde{\boldsymbol{w}})$. We can invoke Lemma 1 on this modified code, repeating as needed to inductively find an optimal solution. Since we have maintained optimality at every stage in the reduction, the code constructed for n symbols is optimal. ■

2.3 Properties of the exponential penalty

2.3.1 Sibling property

As one might expect, certain properties are preserved with nonlinear penalties while others must be modified or discarded. An example of one that must be modified is weighting on internal nodes. As with Huffman coding, because weights are combined in building the exponential Huffman code tree, each internal node in the encoding tree may be said to have a weight equal to its weight at the step of the algorithm in which it is considered a single item. However, weights are not merely summed. For example, the root has weight $F(C_{\mathcal{X}}, \boldsymbol{w}) = \sum_i w_i a^{l_i}$. Any extended binary tree that is a proper subtree of the encoding tree corresponds to an exponential Huffman tree on the leaf weights, although the sum of these weights need not be 1. (Note that the penalties in this dissertation depend on only the depth of the leaf nodes, not on weight of internal nodes, as in [34, 78].)

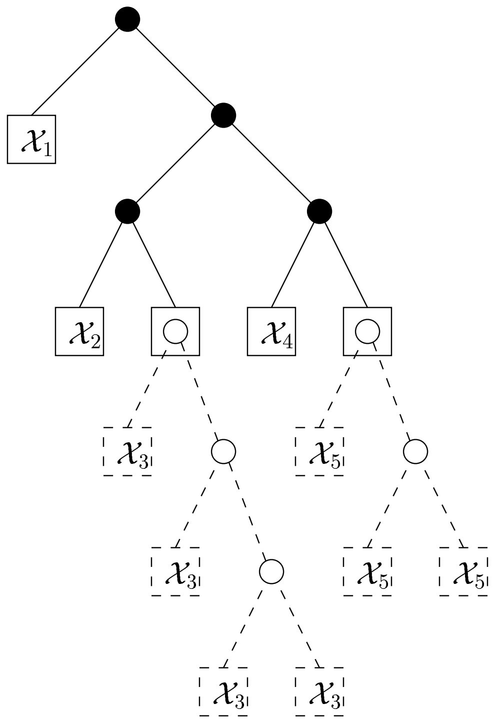


Figure 2.4: Tree representation of a partition of the alphabet $\mathcal{X}$

Definition 6 *A partition of $\mathcal{X}$ into subsets is said to be* compatible *with the code $C_{\mathcal{X}}$ if there exists a subtree of the (main) code tree in which each leaf node has the following property: The corresponding node in the original code tree is an ancestor of all nodes (and only those nodes) representing items in a partition subset of the original tree.*

In Figure 2.4 we illustrate a compatible partition to the code of Figure 2.5(a). This partition consists of five disjoint subsets, $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{X}_3$, $\mathcal{X}_4$, and $\mathcal{X}_5$. Note that the subtree is the same as the tree from Figure 1.1.

A partition is a way of thinking about a code tree in terms of the weighted subtrees with roots at the leaf nodes of the main subtree. Each subtree may be viewed as a code tree itself, and each subtree is optimal if the main tree is optimal. It is easily proved that, given a (potentially infinite) probability distribution with finite entropy, optimality of a code is equivalent to optimality of the codes associated with all its respective compatible subsets [49].

Given the weighting on nodes, we can define a sibling property, an extension of a concept due to Gallager in [29]:

Definition 7 *A binary code tree has the* strong sibling property *if each node, external or internal, except the root, has a sibling and if the nodes can be listed in order of nonincreasing weight with each node being adjacent in the list to its sibling.*

Due to ties, a code tree with the strong sibling property may have nodes listed in nonincreasing weight without siblings being adjacent. However, at least one ordering must have all siblings adjacent. Also note that a code tree for alphabet $\mathcal{X}$ always has $2n-1$ nodes.

Theorem 2 *A binary prefix-free code is an exponential Huffman code (with $a \geq 1$) iff the code tree has the strong sibling property.*

Proof: We use a method similar to that of Gallager. We wish first to prove a binary code tree is an exponential Huffman tree assuming the code tree has the strong sibling property. This is trivial for $n=1$; we may thus use induction and prove it for n assuming it has been proved for $n-1$. First observe that the last two elements in the ordered list must be not only siblings, but also leaf nodes, since an internal node must have a larger weight at least one of its descendents. (Because $a \geq 1$ and $p_i > 0 \; \forall \; i$, both descendents have smaller weight, but this observation allows us to relax this half of the proof to $p_n = 0$ and $a = 1$, a case considered by Gallager, or to $a > \frac{1}{2}$, a case we later consider.) These nodes may thus be siblings in an exponential Huffman code tree. Removing the siblings from the code tree and the two elements from the list results in a reduced code tree with the sibling property and an ordered list for an alphabet of size $n-1$.

We now wish to prove an exponential Huffman code tree has the strong sibling property. Upon execution of the exponential Huffman algorithm, we may generate a list, initially empty, by adding each combined node pair to the start of the list, putting the lesser weight after the greater if they do not have identical weight. Since the weight of the combined node is greater than the weight of the siblings, this

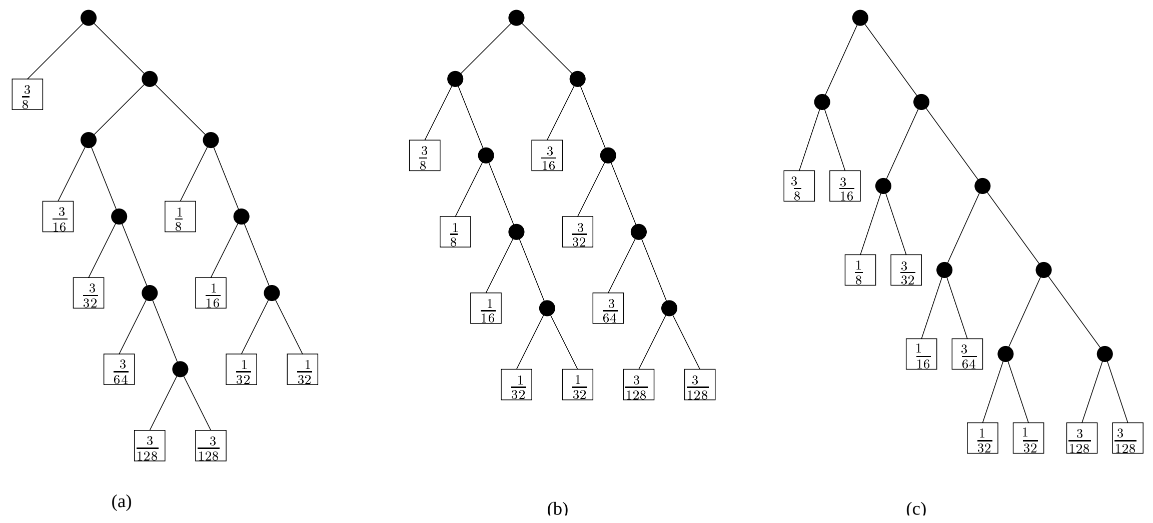


Figure 2.5: Three optimal coding trees, $\boldsymbol{p} = (\frac{3}{8}, \frac{3}{16}, \frac{1}{8}, \frac{3}{32}, \frac{1}{16}, \frac{3}{64}, \frac{1}{32}, \frac{1}{32}, \frac{3}{128}, \frac{3}{128})$

list confirms the sibling property, being a list of nonincreasing weight with adjacent siblings. ■

As previously stated, some optimal codes cannot be created via (exponential) Huffman coding, something that is easy to see via the strong sibling property. Figure 2.5 shows three optimal code trees for $\boldsymbol{p} = (\frac{3}{8}, \frac{3}{16}, \frac{1}{8}, \frac{3}{32}, \frac{1}{16}, \frac{3}{64}, \frac{1}{32}, \frac{1}{32}, \frac{3}{128}, \frac{3}{128})$ and the standard coding problem (solved by the exponential Huffman coding technique for $a = 1$). Tree (a) and tree (b), although corresponding to different length ensembles, are both valid (standard) Huffman coding trees (with item probabilities in leaf nodes). Tree (c) also results in an optimal code, one with the same length ensemble as (b), but is not a Huffman coding tree, lacking the strong sibling property. Tree (c) arises in practice, when, given the length ensemble of (b), one uses the top-down Elias-based method discussed in Section 2.1 to construct an optimal tree. Thus, a practical optimal tree may lack the sibling property. Nevertheless, this property is an effective tool for proving an exponential Huffman tree's optimality.

2.3.2 Interval properties

As previously indicated, taking $a \downarrow 1$, exponential Huffman coding is equivalent to Huffman coding. However, only one of the possibly multiple Huffman length ensembles is produced by exponential Huffman coding at $a = 1 + \epsilon$, where ϵ is an arbitrarily small number greater than 0. We further extend and quantify this in the theorem below.

Theorem 3 *There are only a finite number of $a_i \in (1, +\infty)$ such that $F_{a_i}(L_{\mathcal{X}}, \boldsymbol{p})$ does not have a unique minimizing $L_{\mathcal{X}}$. Consequently, for all $a \in [1, +\infty)$, there exists an $\epsilon_0 > 0$ such that, for all $0 < \epsilon < \epsilon_0$, $F_{a+\epsilon}(L_{\mathcal{X}}, \boldsymbol{p})$ has a unique minimizing $L_{\mathcal{X}}$.*

Proof: With a finite alphabet, $L_{\mathcal{X}}$ may take on only a finite set of values $\{L_{\mathcal{X}}^k\}$ ($k \in \mathcal{S}(m) \triangleq \{1, 2, \ldots, m\}$ for some m) satisfying the Kraft inequality with equality. For a fixed $L_{\mathcal{X}}^k$, $F_a(L_{\mathcal{X}}^k, \boldsymbol{p})$ is a nonzero polynomial in a. Thus, for $j, k \in \mathcal{S}(m)$, $j \neq k$, $F_a(L_{\mathcal{X}}^j, \boldsymbol{p}) - F_a(L_{\mathcal{X}}^k, \boldsymbol{p})$ is also a nonzero polynomial.

Since there are a finite number of difference polynomials, each of which has a finite number of zero-crossings, there may only be a finite number of points at which a tie may occur.

Then at any point $a_0 \in [1, +\infty)$ there exists an $\epsilon_0 > 0$ and a $k_0 \in \mathcal{S}(m)$ such that $F_a(L_{\mathcal{X}}^{k_0}) < F_a(L_{\mathcal{X}}^j)$ for all $j \in \mathcal{S}(m) \backslash \{k_0\}$ and all $a \in (a_0, a_0 + \epsilon_0)$, ■

Thus by specifying a bias toward the positive (or negative), every value of a may correspond to the unique length ensemble of its right (left) ϵ-interval. For $a = 1$, the right interval contains the Huffman code that minimizes variance, as in (2.7). The uniqueness of this minimum variance code was shown by Kou in [55].

A consequence of Theorem 3 is that $\mathbb{R}^+$ consists of open intervals — each of which corresponds to a certain code minimizing $F_b(L_{\mathcal{X}}, \boldsymbol{p})$ — and points separating them, at which ties occur. In other words, if $\mathcal{F}_{\boldsymbol{p}}(b) \triangleq \{L_{\mathcal{X}} \mid F_b(L_{\mathcal{X}}, \boldsymbol{p}) \leq F_b(L'_{\mathcal{X}}, \boldsymbol{p}) \; \forall \; L'_{\mathcal{X}}\}$, then $\mathcal{F}_s p(b)$ is a one-item set for all but a finite number of values of $b \in \mathbb{R}^+$, at which it is larger.

For example, consider $\boldsymbol{p}^{(1)} \triangleq (\frac{4}{9}, \frac{1}{4}, \frac{19}{180}, \frac{1}{10}, \frac{1}{10})$. For the range $b < \lg \frac{5}{4}$, $\mathcal{F}_{\boldsymbol{p}^{(1)}}(b) = \{\{1, 2, 3, 4, 4\}\}$. For $b > \lg \frac{5}{4}$, $\mathcal{F}_{\boldsymbol{p}^{(1)}}(b) = \{\{2, 2, 2, 3, 3\}\}$. At tie point $b = \lg \frac{5}{4}$, both length ensembles are optimal, as is $\{1,3,3,3,3\}$; all three are in $\mathcal{F}_{\boldsymbol{p}^{(1)}}(b)$. In contrast, $\boldsymbol{p}^{(2)} \triangleq (0.58, 0.12, 0.11, 0.1, 0.09)$ has no three-way ties, but does have a transitional (two-way) tie at $b = \lg \frac{58}{23}$.

Using $1+\epsilon$ provides us with an alternate proof of Huffman coding optimality, but one which applies only to the minimum variance length ensemble. This is also an alternative method of Huffman coding; by keeping track of ϵ's, one can produce an optimal Huffman code with minimum variance among such codes. We term this *epsilon Huffman coding.*

This coding method is equivalent to others found throughout the literature. The first occurrence of this is also the simplest, called *bottom-merge Huffman coding*, introduced by Schwartz in [84]. It yields a code $C_{\mathcal{X}}$ with the aforementioned unique $L_{\mathcal{X}}$. At the time, this was noted to simultaneously minimize two values among optimal codes. First, of the optimal trees, the bottom-merge code has minimal maximal length. Second, it has minimal sum of lengths, $\sum_i l_i$, a measure of tree flatness (balance). Note that while the nondeterministic Huffman algorithm may yield trees (a) and (b) in Figure 2.5, only the tree in (b) is found using bottom-merge or epsilon Huffman coding.

Practical implementation of bottom-merge coding is done in linear time with two queues — one for singletons, one for combined items — both ordered by probability. Given ties in probabilities, the bottom-merge method prefers merging singletons before combined items. Since combined items are then put at the end of the combined item queue in order of formation, the queue is secondarily ordered by maximum length, and we thus minimize maximum length via this bottom-merge two-queue method [91].

By preferring combined items, one may obtain a *top-merge code*, one which maximizes the previously discussed values instead of minimizing them. Examples of both are in Figure 2.6, in which we consider the status of Huffman coding for $\boldsymbol{p} = (\frac{3}{8}, \frac{3}{16}, \frac{1}{8}, \frac{3}{32}, \frac{1}{16}, \frac{3}{64}, \frac{1}{32}, \frac{1}{32}, \frac{3}{128}, \frac{3}{128})$ in the penultimate step of the algorithm. Previous to this step, the collection of combined trees consists of the three trees shown in (a) — one consisting of a single item, one consisting of five items (represented by either the tree itself or the $\frac{3}{8}$ circle) and one consisting of four items (represented by

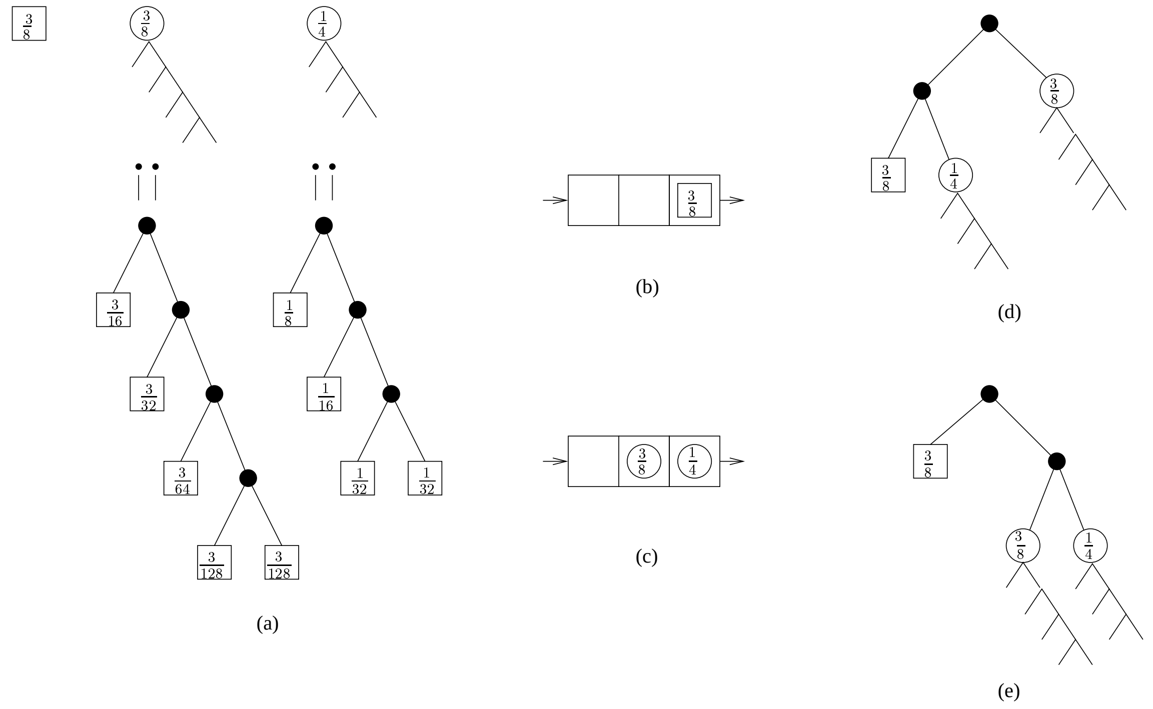


Figure 2.6: Bottom-merging versus top-merging

either the tree itself or the $\frac{1}{4}$ circle). The singleton queue is (b) and the combined item queue is (c). If a bottom-merge algorithm is used, the $\frac{3}{8}$ singleton is merged with the $\frac{1}{4}$ tree, and the tree in (d) (or Figure 2.5(b)) results after the algorithm terminates. If a top-merge algorithm is used, the $\frac{3}{8}$ five-item tree is merged with the $\frac{1}{4}$ tree, and the tree in (e) (Figure 2.5(a)) results.

The bottom-merge results of Schwartz have been further extended. Depending on context, the code has been called a *minimum variance Huffman code* [55], *best Huffman tree* [68], *compact Huffman tree* [38], or *minimal Huffman tree* [27]. It was also considered algebraically by Knuth in [53]. In all cases, the resulting output is the unique length ensemble that, of the possible Huffman codes, minimizes all of a number of functions including: all weighted moments $\sum_i p_i |l_i - c|^\nu$ for $c \in \mathbb{R}$, $\nu > 1$ (including variance); all unweighted moments about zero $\sum_i l_i^\nu$ for $\nu > 0$ (including sum of and maximum of lengths); and any convex function of the lengths (including, of course, $F_b(L_{\mathcal{X}}, \boldsymbol{p})$) [27, 68].

The bottom-merge code is also always the lexicographically first of the optimal set of Huffman length ensembles (when lengths are sorted in decreasing order). We return to this topic later; more can be found in the cited papers. (Note, however, that many papers wrongly attribute to Tamaki [87] the connection between epsilon Huffman coding and Huffman coding for minimum variance — which may be easily seen by the expansion of the first equation in (2.7). While Tamaki's observations on optimal trees are noteworthy, they do not include the aforementioned connection, noted in [53, 55, 78], among others.)

2.4 Another exponential penalty

If $0 < a < 1$ ($b < 0$), minimization of $\sum_i p_i a^{l_i}$ is not an interesting goal, but maximization is. The tools for maximization of this value have been largely seen as a side effect of the tools for minimization for $a > 1$ [45]. However, there are situations in which one might wish a penalty of this form.

2.4.1 Motivation

Consider the following problem: We have an indefinite open window of time in which to transmit a message at constant bitrate. We will not know the duration of this window until it closes, and only a complete message codeword is useful. The period could be a brief period of contact to a remote device usually out of range, such as a mobile phone or remote exploration device. Alternatively, it could be a period of transmission before an adversary cuts communication lines or ends a siege, as in Rényi's telling of the story of the Bar Kochba revolt.

Assume the duration of the window of opportunity is independent of the communicated message, X, and is memoryless. Memorylessness implies that the window duration is distributed exponentially. Therefore, if we quantize the time in terms of bits we may send within our window, T, this random variable is geometrically distributed, that is,

$$P\{T = t\} = (1-a)a^t,\ t = 0, 1, 2, \ldots \tag{2.31}$$

with known parameter $a < 1$. We then wish to maximize the following probability:

$$\begin{aligned}
P\{l(X) \leq T\} &= \sum_{t=0}^{\infty} P\{T = t\} \cdot P\{l(X) \leq t\} && (2.32)\\
&= \sum_{t=0}^{\infty} (1-a)a^t \cdot \sum_{i=1}^{n} p_i 1_{l_i \leq t} && (2.33)\\
&= \sum_{i=1}^{n} p_i \cdot (1-a) \sum_{t=l_i}^{\infty} a^t && (2.34)\\
&= \sum_{i=1}^{n} p_i a^{l_i} \cdot (1-a) \sum_{t=0}^{\infty} a^t && (2.35)\\
&= \sum_{i=1}^{n} p_i a^{l_i}. && (2.36)
\end{aligned}$$

Thus this is the expectation of a utility function — which may be viewed as negative cost — decaying exponentially with time (or length). In this case, then, we wish to maximize the expectation of a decaying exponential instead of minimizing the expectation of a growing exponential. These two problems may at first seem different, but their solutions are similar.

2.4.2 Problem structure and solution

The utility function here is $f(l) = a^l = 2^{bl}$ for $a \in (0, 1)$ ($b \in (-\infty, 0)$). Then we wish to maximize $F(L_{\mathcal{X}}, \boldsymbol{p}) = E_{\boldsymbol{p}}[f(l(X))]$ subject to the Kraft inequality. Equivalently, we wish to minimize cost function $F^{-}(L_{\mathcal{X}}, \boldsymbol{p}) \triangleq -F(L_{\mathcal{X}}, \boldsymbol{p})$ (minimizing as in (2.2)) or $G(L_{\mathcal{X}}, \boldsymbol{p})$ (minimizing quasilinear Campbell formulation as in (2.3)). Note that the latter average is that used by risk-seeking optimization algorithms [94].

Before solving this problem, we should first note and emphasize that this formulation is not the minimization of a convex function. Still, it has many properties in common with the previously considered penalty, as well as with standard Huffman coding.

If exponential parameter $b \leq -1$, the real-valued problem has no maximum $L_{\mathcal{X}}^{\dagger}$ in $\mathbb{R}^{|\mathcal{X}|}$, as $\sup_{L_{\mathcal{X}}} F(L_{\mathcal{X}}, \boldsymbol{p}) = p_1$, which is maximized by $L_{\mathcal{X}}^{+\infty} \triangleq \{0, +\infty, \cdots, +\infty\}$. This implies that, in some sense, it is best not to transmit but merely to assume the maximum likelihood solution for such extreme utility functions. In the case of the fortress, if we alter the problem slightly to allow action without a received message, it is optimal to defend the fortress as though the most likely event is the one that is occurring. However, in the problem formulation we wish to know a message with certainty, and thus this is not practical.

Note that, if $b = -1$ and $p_1 = p_2$, any combination of lengths for the tied maximum probabilities satisfying the Kraft inequality with equality yields $\sup_{L_{\mathcal{X}}} F(L_{\mathcal{X}}, \boldsymbol{p}) = p_1$. Thus there is no unique value, finite or otherwise, for $L_{\mathcal{X}}^{\dagger}$ in this situation.

However, for $b \in (-1, 0)$, a more realistic range of parameters, equations (2.8–2.17) still apply if we replace J with $-J^{-}$ and λ with $-\lambda^{-}$. Thus, the solutions remain the same, but, for example, (2.13) becomes

$$\lambda^{-} = -b \cdot \left(\sum_j p_j{}^{\frac{1}{1+b}} \right)^{1+b} \geq 0. \qquad (2.37)$$

This type of exponential penalty problem is not a convex problem, but we nevertheless know the real-valued solution of the form in (2.14–2.15) is optimal because the Karush-Kuhn-Tucker conditions are satisfied, and $F^{-}(L_{\mathcal{X}}, \boldsymbol{p})$ (or $G(L_{\mathcal{X}}, \boldsymbol{p})$) is thus minimized. The expressions for the objective function at this value (2.16) and the bounds (2.21) for the optimal integer solution, $L_{\mathcal{X}}^{*}$, remain the same as well. These bounds in terms of the example of maximizing probability of message communication in Section 2.4.1 are

$$2^{bH_{\alpha}(\boldsymbol{p})+1} < \max_{L_{\mathcal{X}}} P\{l(X) \leq T\} \leq 2^{bH_{\alpha}(\boldsymbol{p})} \qquad (2.38)$$

where we recall that $\alpha = \frac{1}{1+b}$ and $a = 2^b \in (0.5, 1)$ is the parameter of the geometric distribution of T. As before, equality holds iff the ideal solution has all integer lengths. Thus Rényi's siege dilemma, appropriately, has a solution connected to Rényi entropy.

For $b \leq -1$, the analogous bounds are obtained by substituting $L_{\mathcal{X}}^{+\infty}$ into (2.20), yielding $\frac{1}{b} \lg p_1 \leq G(L_{\mathcal{X}}^*) < \frac{1}{b} \lg p_1 + 1$ and $p_1 2^b < F(L_{\mathcal{X}}^*) \leq p_1$, analogous to (2.21) and (2.22), respectively.

For example, if $\boldsymbol{p} = (\frac{4}{9}, \frac{1}{4}, \frac{19}{180}, \frac{1}{10}, \frac{1}{10})$ and $b \leq -1$, $\frac{2^{b+2}}{9} < F(L_{\mathcal{X}}^*) \leq \frac{4}{9}$. For all $b < 0$, Lemma 1 still applies (to $F^-(L_{\mathcal{X}}, \boldsymbol{p})$ or $G(L_{\mathcal{X}}, \boldsymbol{p})$) since it only assumes monotonicity.

All the assumptions of Theorem 1 still hold as well — the only change is that equations (2.26–2.30) prove a maximization (of $F(C_{\mathcal{X}}, \boldsymbol{w})$) while Lemma 1 proves minimization. Thus exponential Huffman coding works for this penalty, and, combining with our previous results:

Theorem 4 *The exponential Huffman algorithm is optimal for all $b \in \mathbb{R}$ ($a > 0$). For $b = 0$, it is Huffman coding, $b > 0$ was proved in Theorem 1, and, for $b < 0$, if C^* is an exponential Huffman code for parameter b, and C is any other code, then $F_b^-(C^*, \boldsymbol{p}) \leq F_b^-(C, \boldsymbol{p})$, i.e., $F_b(C^*, \boldsymbol{p}) \geq F_b(C, \boldsymbol{p})$, and thus exponential Huffman coding for $b < 0$ maximizes expected utility.*

2.5 Properties of both exponential penalties

2.5.1 Sibling property

Although the above theorem extends to this new exponential case, Theorem 2 no longer holds, that is, the strong sibling property is no longer necessary to have an exponential Huffman tree. A trivial counterexample is $a = \frac{2}{3}$ and $\boldsymbol{p} = (\frac{3}{4}, \frac{1}{4})$. However, this property is sufficient for $a > \frac{1}{2}$ ($b > -1$), implying an exponential Huffman tree when it does occur, as noted in the parenthetical comment in the proof to Theorem 2.

Given an exponential Huffman tree, the following weaker property holds:

Definition 8 *A binary code tree has the* weak sibling property *if each node (except the root) has a sibling, if the leaf nodes can be listed in order of nonincreasing weight with each node being adjacent in the list to its sibling if the sibling is also a leaf node, and if every extended subtree has the identical property with its associated subset.*

This weakened property is a direct result of the exponential Huffman coding procedure. In the case of $a \geq 1$, the weak sibling property, the strong sibling property, and exponential Huffman coding are equivalent. (Exponential Huffman coding implies the weak property, the weak property can be applied to the height-truncated version of the optimal tree to obtain the strong property, and the strong property was previously shown to imply Huffman coding.)

2.5.2 Properties on the real line

For any $\boldsymbol{p}$, if $a < \frac{1}{2}$ $(b < -1)$, the unary length ensemble, $L_{\mathcal{X}} = \{1, 2, \cdots, n-1, n-1\}$, is the maximizing argument, which can be seen constructively using the fact that, in exponential Huffman coding for these values, combined weights are always less than the maximum of the two original weights. Thus each step merely builds up the maximal height tree.

Theorem 3 still holds for $b \in (-\infty, 0)$ $(a \in (0,1))$ so now we may view the entire real line as consisting of open intervals — each of which corresponds to a certain code optimizing $F_b(L_{\mathcal{X}}, \boldsymbol{p})$ — and points at which a tie occurs. We may thus extend the definition of $\mathcal{F}_{\boldsymbol{p}}(b)$ to $b \leq 0$:

$$\mathcal{F}_{\boldsymbol{p}}(b) \triangleq \begin{cases} \{L_{\mathcal{X}} \in \mathcal{L}_{|\mathcal{X}|} \text{ such that } F_b(L_{\mathcal{X}}, \boldsymbol{p}) \leq F_b(L'_{\mathcal{X}}, \boldsymbol{p}) \ \forall\ L'_{\mathcal{X}} \in \mathcal{L}_{|\mathcal{X}|}\}, & b > 0 \\ \{L_{\mathcal{X}} \in \mathcal{L}_{|\mathcal{X}|} \text{ such that } E_{\boldsymbol{p}}[l(X)] \leq E_{\boldsymbol{p}}[l'(X)] \ \forall\ L'_{\mathcal{X}} \in \mathcal{L}_{|\mathcal{X}|}\}, & b = 0 \\ \{L_{\mathcal{X}} \in \mathcal{L}_{|\mathcal{X}|} \text{ such that } F_b(L_{\mathcal{X}}, \boldsymbol{p}) \geq F_b(L'_{\mathcal{X}}, \boldsymbol{p}) \ \forall\ L'_{\mathcal{X}} \in \mathcal{L}_{|\mathcal{X}|}\}, & b < 0 \end{cases} \tag{2.39}$$

where we recall that $\mathcal{L}_{|\mathcal{X}|}$ is the set of length ensembles satisfying the Kraft inequality with equality. From the definition of $G_b(L_{\mathcal{X}}, \boldsymbol{p})$ given in (2.6), this is equivalent to

$$\mathcal{F}_{\boldsymbol{p}}(b) \triangleq \{L_{\mathcal{X}} \in \mathcal{L}_{|\mathcal{X}|} \text{ such that } G_b(L_{\mathcal{X}}, \boldsymbol{p}) \leq G_b(L'_{\mathcal{X}}, \boldsymbol{p}) \ \forall\ L'_{\mathcal{X}} \in \mathcal{L}_{|\mathcal{X}|}\}. \tag{2.40}$$

For all but a finite set of points $b \in \mathbb{R}$, $|\mathcal{F}_{\boldsymbol{p}}(b)| = 1$; on this set of points, there is at least one tie, i.e., $|\mathcal{F}_{\boldsymbol{p}}(b)| > 1$.

Taking $b \uparrow 0$ ($a \uparrow 1$) yields an *inverse epsilon Huffman coding* technique that breaks ties by favoring tree codes with maximal variance, since the $O(b)$ term in the first equation of (2.7) is now negative. (This is identical to top-merge coding, previously discussed.) If there is a unique Huffman length ensemble $L_{\mathcal{X}}^{hu}$, it may be obtained using either method, being the unique solution of

$$L_{\mathcal{X}}^{hu} = \lim_{b \to 0} \left[\arg\min_{L_{\mathcal{X}}} \{G_b(L_{\mathcal{X}}, \boldsymbol{p})\} \right] = \lim_{b \to 0} \left[\arg\min_{\substack{\sum_i 2^{-l_i} \leq 1, \\ l_i \in \mathbb{N}}} \{\mathrm{sgn}(b) \cdot E_{\boldsymbol{p}}[2^{bl(X)}]\} \right] \tag{2.41}$$

where $\mathrm{sgn}(b)$ denotes the signum of b:

$$\mathrm{sgn}(x) \triangleq \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \tag{2.42}$$

The discontinuous signum function emphasizes that the exponential penalty problem may be properly viewed as two distinct problems, one for $b > 0$ and one for $b < 0$. Nevertheless, as shown with equation (2.40), there is continuity; both positive and negative have ideal solutions and actual algorithms of the same form, and, for $b \to 0$, both converge to optimal coding. Because of this solution, trees get more "flat" with increasing b and less "flat" with decreasing b, a behavior expected for either problem, and thus not surprising for the composite. We quantify this in Section 2.5.3.

For the previous example where $\boldsymbol{p}^{(1)} = (\frac{4}{9}, \frac{1}{4}, \frac{19}{180}, \frac{1}{10}, \frac{1}{10})$, $\mathcal{F}_{\boldsymbol{p}^{(1)}}(b) = \{\{1,2,3,4,4\}\}$ $\forall\, b < \lg \frac{5}{4}$, with other b's as previously specified. For $\boldsymbol{p}^{(2)} = (0.58, 0.12, 0.11, 0.1, 0.09)$, $\mathcal{F}_{\boldsymbol{p}^{(2)}}(b) = \{\{1,2,3,4,4\}\}$ $\forall\, b < \lg \frac{5}{9}$, $\mathcal{F}_{\boldsymbol{p}^{(2)}}(b) = \{\{1,3,3,3,3\}\}$ $\forall\, b \in (\lg \frac{5}{9}, \lg \frac{58}{23})$ and $\mathcal{F}_{\boldsymbol{p}^{(2)}}(b) = \{\{2,2,2,3,3\}\}$ $\forall\, b > \lg \frac{58}{23}$. The transition values have two-way ties. This is illustrated in Figure 2.7, where the top range represents the trees optimal for $\boldsymbol{p}^{(1)} = (\frac{4}{9}, \frac{1}{4}, \frac{19}{180}, \frac{1}{10}, \frac{1}{10})$ and the bottom those for $\boldsymbol{p}^{(2)} = (0.58, 0.12, 0.11, 0.1, 0.09)$.

For implementation purposes, it is noteworthy that the two-queue approach to Huffman coding extends directly to all values of $b \in \mathbb{R}$ for exponential Huffman coding. Both queues remain ordered since the weight-combining method is linear.

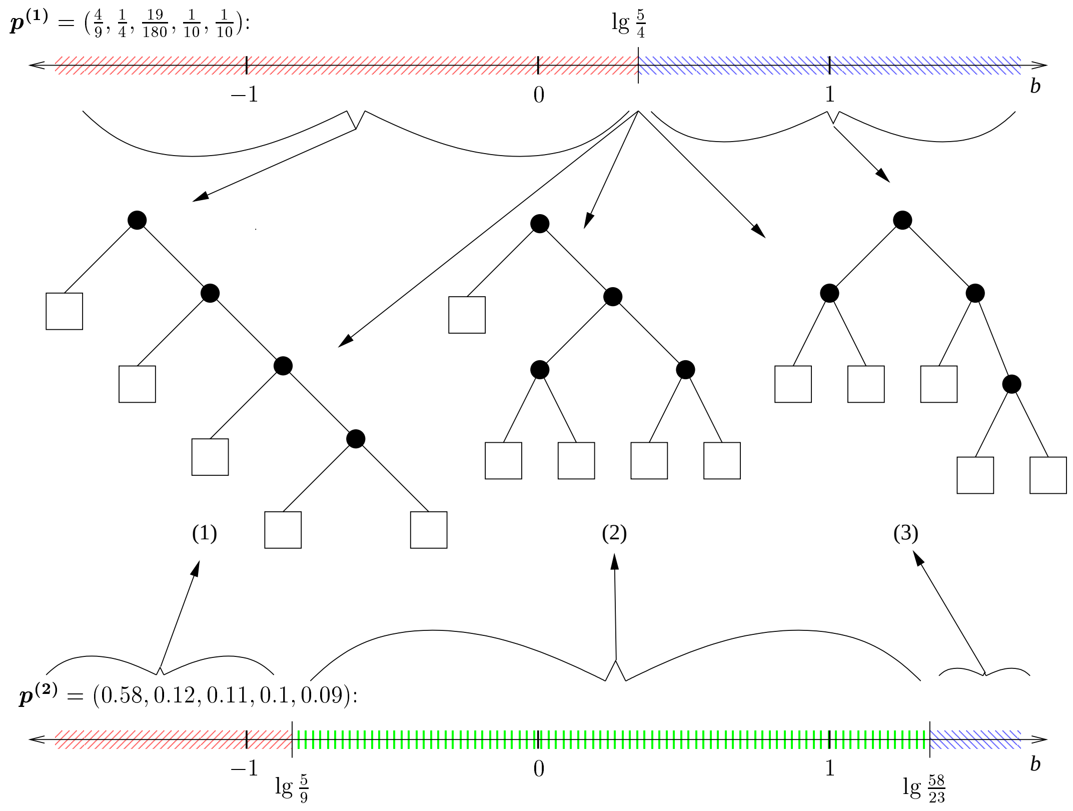

$\boldsymbol{p}^{(\mathbf{1})} = (\frac{4}{9}, \frac{1}{4}, \frac{19}{180}, \frac{1}{10}, \frac{1}{10})$:
$\lg \frac{5}{4}$
-1
0
1
b
(1)
(2)
(3)
$\boldsymbol{p}^{(\mathbf{2})} = (0.58, 0.12, 0.11, 0.1, 0.09)$:
-1
$\lg \frac{5}{9}$
0
1
$\lg \frac{58}{23}$
b

Figure 2.7: Optimal code trees for two different distributions over the range of $b \in \mathbb{R}$

The smallest item is always at the head of one of the two queues, and, with sorted weights, exponential Huffman coding has a linear time algorithm. An example for $b = 0$ is shown in Figure 2.6. (Note that while this ordering seems counterintuitive for $b < -1$, it holds because the combined items queue never contains more than one item in this range of parameters. Thus a unary tree is always created.)

2.5.3 Interval properties

As with bottom-merge Huffman coding, using a bottom-merge technique here yields a deterministic algorithm which, for a given mass function, returns the minimum code among optimal codes, thus minimizing maximal length among such codes. To prove this, we need the following definitions and lemmas patterned after those of Forst and Thorup in [27]. We later show the bottom-merge length ensemble for a and $\boldsymbol{p}$ is equivalent to the unique length ensemble for $a + \epsilon$ and $\boldsymbol{p}$.

Definition 9 *If i and j represent trees (as do, for example, codes, nodes or length ensembles), length ensembles $L(i)$ and $L(j)$ have the relation $L(i) > L(j)$ if j is lexicographically smaller than i (when lengths are sorted in lexicographical order).* Addition *for such ensembles refers to formation of a merged tree — one takes the union of the sets and adds 1 to each element. Thus, for example, if $i = \{2, 2, 1\}$ and $j = \{1, 1\}$, then $L(i) + L(j) = \{3, 3, 2, 2, 2\}$. Denote the weight of the root node of (sub)tree i as $w(i)$. Note that leaf nodes are trivial subtrees. Then define* enhanced weight *w' to have the relation $w'(i) > w'(j)$ if either $w(i) > w(j)$, or $w(i) = w(j)$ and $L(i) > L(j)$. Addition for such enhanced weights also represents a merged tree — one adds their length ensembles and uses weight $w(k) = a \cdot (w(i) + w(j))$. If $\mathcal{W}$ represents the set of (leaf node) weights, a $\mathcal{W}$* tree *is one with weights $\mathcal{W}$ that is Huffman-constructible. Finally, $h(i) \triangleq \max_{l \in L(i)} l$ is the* height, *or maximal length codeword, associated with tree i.*

Lemma 2 *Let $\mathcal{W}$ be a set of weights with $|\mathcal{W}| \geq 2$. Suppose T_0 is a minimal exponential Huffman tree, that is, the optimal tree that is minimum lexicographically. We know there are at least two items corresponding to leaf nodes at level $h(T_0)$; denote* *them u and v. Suppose $\mathcal{W}'$ denotes the set of weights with items u and v replaced by item x of weight $w(x) = a \cdot (w(u)+w(v))$. Let T_1 denote the $\mathcal{W}'$ tree which is T_0 with u and v nodes pruned (deleted), their parent node becoming the leaf node corresponding to item x. If there exists a distinct optimal exponential Huffman tree T_2 for $\mathcal{W}'$ such that every item of weight $w(x)$ has level at least $h(T_0)$, then $L(T_2) > L(T_1)$.*

Proof: We have $h(T_2) \geq h(T_0) \geq h(T_1)$. Unless all are equalities, $L(T_2) > L(T_1)$, which is what we set out to prove. We may thus assume $h(T_2) = h(T_0) = h(T_1)$. Call this height h and the number of leaf nodes in T_1 at level h, n_h.

Note first that this implies T_1 is an optimal tree for $\mathcal{W}'$ with x at level $h-1$ of a tree of height h. Then, due to this optimality, $w(x)$ cannot be smaller than all other weights in $\mathcal{W}'$. Thus $w(x) \geq \max(w(u), w(v))$, since u and v have the lowest weights in $\mathcal{W}$.

Refer to the ith leaf node of T_1 at lowest level h as $\Xi_i(T_1)$. For each such leaf node, $w(\Xi_i(T_1)) \leq w(x)$, since x is at level $h-1$. (This is due to Lemma 1; recall that this still holds for the leaf nodes of exponential Huffman trees.) But in T_2, x is on level h since we asserted in our conditions it can be no less and, since $h = h(T_2)$, it can be no more. Thus T_2 has more leaf nodes than T_1 at level h, and is therefore lexicographically greater. ■

Lemma 3 *Let $\mathcal{W}$ and $\mathcal{W}'$ represent weights as in the previous lemma. Suppose T^* and T'^* are (exponential) Huffman trees for $\mathcal{W}$ and $\mathcal{W}'$, respectively. Suppose further that T'^* is minimal and in T^* all leaf nodes of weight $w(x) = a \cdot (w(u) + w(v))$ are at levels $h(T^*) - 1$ or $h(T^*)$. Then T^* is minimal as well.*

Proof: Let T be an arbitrary (exponential) Huffman tree for $\mathcal{W}$. Then there are nodes of weight $w(u)$ and $w(v)$ that are siblings and may thus be pruned to form a new tree, T', optimal for $\mathcal{W}'$. $L(T'^*) \leq L(T')$ due to T'^* being minimal. The contrapositive of the previous lemma implies that there is a leaf node of weight $w(x)$ at level strictly less than $h(T)$ in T'^*. Therefore, since all leaf nodes of weight $w(x)$ in T^* are at levels $h(T^*) - 1$ or $h(T^*)$, $L(T^*) \leq L(T)$ for all such T, and T^* is minimal. ■

Lemma 4 *Suppose i, j, k represent trees and $L(i) < L(j)$. Then $L(i) + L(k) < L(j) + L(k)$.*

Proof: Let $\xi(L(i), r)$ represent the rth entry of $L(i)$. Then, if $L(i) < L(j)$, there exists a positive integer s such that $\xi(L(i), r) = \xi(L(j), r)\ \forall\ r < s$ and $\xi(L(i), s) < \xi(L(j), s)$. Let $\pi = \xi^{-1}(L(k), \lambda)$ denote the last position in $L(k)$ for which $\xi(L(k), \pi) \geq \lambda$, or 0 if $\xi(L(k), 1) < \lambda$. Then $L(i) + L(k)$ and $L(j) + L(k)$ first differ at position $\pi^* = s + \xi^{-1}(L(k), \xi(L(j), s))$, with $\xi(L(i)+L(k), \pi^*) = 1 + \xi(L(i), s) < 1 + \xi(L(j), s) = \xi(L(j) + L(k), \pi^*)$ and thus $L(i) < L(j)$. ■

We can now prove the theorem.

Theorem 5 *The bottom-merge exponential Huffman algorithm returns the minimum length ensemble among all optimal codes.*

Proof: Recall that the bottom-merge algorithm uses two queues. Without loss of generality, we consider the two queues to be one, and the combined tree weight, instead of going to the end of the combined tree queue, is inserted just before the first singleton of greater weight, or at the end of the queue if there is no such singleton. It is easily seen that this is equivalent to the bottom-merge case.

This induces an enhanced weight ordering w' on items, which is trivial initially ($w' = \{w, \{0\}\}$). We wish to show inductively that the minimal optimal tree is always created.

We first wish to show how items representing trees combine. Denote by $q_i^{(n)}$ the kth item in the queue (relative to the head) at step n. Ties occur only when $w(q_2^{(n)}) = w(q_3^{(n)})$, so we may assume this to be the case. We consider two cases according to whether or not $w(q_1^{(n)}) = w(q_2^{(n)})$. If they are not equal, then the minimal tree of those with weight $a \cdot (w(q_1^{(n)}) + w(q_2^{(n)}))$ is the one formed by combining $q_1^{(n)}$ and $q_2^{(n)}$, due to Lemma 4. If they are equal, then the minimal tree is similarly obtained for like reasons unless $w(q_1^{(n)}) = w(q_2^{(n)}) = w(q_3^{(n)}) = w(q_4^{(n)})$. If this is the case, then suppose we take items $q_u^{(n)}$ and $q_v^{(n)}$. Then $L(q_u^{(n)}) + L(q_v^{(n)}) \geq L(q_1^{(n)}) + L(q_v^{(n)}) \geq L(q_1^{(n)}) + L(q_2^{(n)})$, so taking the first two is always minimal among combined items of that weight.

We know that the resulting tree T is an optimal tree, so we need to prove that the above implies it has minimal weight. Inductively assume subtree T' for weights

$\mathcal{W}'$ (as defined in Lemma 3 with x, u, and v) is minimal. (The base case is trivial.) Note that, in T', if $w(j) = w(k)$ and $L(j) > L(k)$, then $l_j \leq l_k$. If these were not the case, switching the node positions would result in a smaller value for $L(T')$, which is minimal. Thus, no leaf node of weight $w(x)$ may be on a level less than $h(T) - 1$ since $L(x) > \{0\}$, the length ensemble for a singleton. Thus, from Lemma 3, T must be minimal as well. ■

We next wish to prove that bottom-merge exponential Huffman coding for a is equivalent, for some sufficiently small ϵ_0, to exponential Huffman coding for $a + \epsilon$ for any $\epsilon \in (0, \epsilon_0)$. This has already been shown for $a = 1$ by Forst and Thorup in [27], so here we consider only cases $a > 1$ and $0 < a < 1$. Note first that

$$\sum_i w_i (a+\epsilon)^{l_i} = \sum_i w_i a^{l_i} + \epsilon \sum_i w_i l_i a^{l_i - 1} + O(\epsilon^2) \qquad (2.43)$$

so the $a + \epsilon$ length ensemble is optimal for a and minimizes $\mathrm{sgn}(\lg a) \cdot \sum_i w_i l_i a^{l_i - 1}$ among a-optimal solutions. In other words, if $f_1(x, y) \triangleq y a^x$ and $f_2(x, y) \triangleq y x a^{x-1}$, then $(a+\epsilon)$-exponential Huffman coding minimizes $\mathrm{sgn}(\lg a) \cdot \sum_i f_1(l_i, w_i)$ and, among such optimal ensembles, finds the ensemble such that, if $\ell_\psi \triangleq \sum_i \psi(l_i, w_i)$ for $\psi = \mathrm{sgn}(\lg a) \cdot f_2(x, y)$, ℓ_ψ is minimized. Following [27], let us define

$$\delta_{\psi,v,w,a}(k) \triangleq \psi(k+1, v) + \psi(k+1, w) - \psi(k, a \cdot (v + w)). \qquad (2.44)$$

Based on f_2,

$$\begin{aligned} \mathrm{sgn}(\lg a) \cdot \delta_{f_2,v,w,a}(k) &= f_2(k+1, v) + f_2(k+1, w) - f_2(k, a \cdot (v+w)) && (2.45) \\ &= -(v+w) k a^k + v(k+1) a^k + w(k+1) a^k && (2.46) \\ &= (v+w) a^k. && (2.47) \end{aligned}$$

For a tree T with t leaf nodes of weight $\boldsymbol{w}^{\{T\}}$ (enumerated such that $w_t^{\{T\}}$ is the smallest weight), further define $\delta_{\psi,a}(T) \triangleq \delta_{\psi, w_t^{\{T\}}, w_{t-1}^{\{T\}}, a}(h(T) - 1)$. We may thus show the following:

Theorem 6 *If $a \neq 1$, the minimum Huffman tree minimizes $\sum_i \psi(l_i, w_i)$ among optimal Huffman trees for any ψ that, like $\mathrm{sgn}(\lg a) \cdot f_2(x, y)$, has $\delta_{\psi,v,w,a}(k)$ increasing in k. Consequently, since this was already shown for $a = 1$ in [27], the bottom-merge length ensemble for all $a \in \mathbb{R}$ is equivalent to the unique length ensemble for $a + \epsilon$.*

Proof: This may be shown via induction on the tree size, since this implication trivially holds for a tree with two items. We may thus assume it holds for $\mathcal{W}'$, the set of weights introduced in Lemma 2. Given a nonminimum $\mathcal{W}'$-Huffman tree Q' and minimum Huffman tree S', we thus have $\ell_\psi(S') \leq \ell_\psi(Q')$. (If there are no other $\mathcal{W}'$-Huffman trees, then we are done.) Note also that, because the smallest two nodes cannot be any higher than in S, $\delta_{\psi,a}(S) \leq \delta_{\psi,a}(Q)$.

Thus, combining this with the minimal height of minimal trees:

$$\begin{aligned} \ell_\psi(S) &= \delta_{\psi,a}(S) + \ell_\psi(S') && (2.48)\\ &\leq \delta_{\psi,a}(Q) + \ell_\psi(Q') && (2.49)\\ &= \ell_\psi(Q). && (2.50) \end{aligned}$$

The equalities are due to the definition of $\delta_{\psi,a}(k)$. Thus, the bottom-merge tree is optimal for all secondary $\psi(x, y)$ for which $\delta_{\psi,a}(k)$ is increasing. This argument may be repeated for strictly increasing $\delta_{\psi,a}(k)$ using strict inequality between distinct trees to show uniqueness of this ψ-minimum result. Take $\psi(x, y) = \mathrm{sgn}(\lg a) \cdot f_2(x, y) = \mathrm{sgn}(\lg a) \cdot yxa^{x-1}$. Since $\delta_{f_2,v,w,a}(k)$ as in (2.47) is strictly increasing, $(a+\epsilon)$-exponential Huffman coding is identical to bottom-merge Huffman coding. ■

This result illuminates the structure of the problem in a more specific manner than Theorem 3; the intervals corresponding to ranges of solutions must be lexicographically nonincreasing. Using methods parallel to those used in Lemmas 2–4 and Theorems 5–6, it may be shown that tie points have an interval corresponding to their minimum exponential Huffman tree to their right and an interval corresponding to their maximum exponential Huffman tree to their left.

Thus, with increasing b, the optimal tree is "flattened" in terms of lexicographical order. Figure 2.7 has two previously discussed examples of this. More specifically, if a given probability mass function has identical exponential Huffman trees for $a = a_1$ and $a = a_2$, then the tree is optimal for all points in the interval defined by the two points. Another result of this property for the left interval is the uniqueness of the maximum variance Huffman code, a code identical to the top-merge Huffman code.

If limited by length, one may wish to find a reasonably low value of b sufficiently high so $l_n \leq l_{max}$ for some l_{max}. Due to the above properties, such a value may be found via binary search over b. The code obtained is not necessarily the same as the optimal code for $b = 0$ limited to the set of codes with this property. The latter problem, previously solved, is one we consider in Chapter 4 and one that can be formulated (4.24) as with other problems in this chapter.

As a side note, we mention that we can easily extend the exponential Huffman algorithm to D-ary alphabets in the same manner as for Huffman coding.

2.6 Extending to other penalties

One might be tempted to see whether a generalization of the Huffman method extends to other expectation penalties beyond the linear and exponential cases. We briefly present an intuitive sketch for why, in some sense, it cannot, with citations to previous work with more involved discussions on this topic.

Let us first extend the definition of $f(l) : \mathbb{N} \to \mathbb{R}^+$ to $f(l) : \mathbb{R}^+ \to \mathbb{R}^+$. We may assume, without loss of generality, that f is a monotonic function with no zero-crossings which is in the set $\mathcal{C}^2$, i.e., the first two derivatives (f', f'') exist and are continuous at all points. In addition, let us assume that, because values at $l \in \mathbb{R}^+ \backslash \mathbb{N}$ are not evaluated, any equalities found for all $l \in \mathbb{N}$ may be assumed to apply over all $l > 0$ without loss of generality. (Without this assumption, a piecewise smooth polynomial approximation of the exponential would work in spite of it being only a trivial extension.)

Let $L_{\tilde{\mathcal{X}}}$ denote the lengths of the optimal alphabet of the code formed after the first step of coding via the Huffman method. We then need $F(L_{\mathcal{X}}, \boldsymbol{p}) - F(L_{\tilde{\mathcal{X}}}, \boldsymbol{p})$ to be independent of $L_{\mathcal{X}}$ in order to yield a set of equalities like (2.26–2.30). We also assume that weights are added upon merging, followed by a functional transformation ϕ, that is, $\phi(w_j, w_k) = \phi(w_j + w_k)$. This seems necessary for symmetry, and, although it is actually not [39], this is a good first approximation.

With the above assumptions, we find

$$\begin{aligned}
& & F(L_{\mathcal{X}}, \boldsymbol{p}) - F(L_{\tilde{\mathcal{X}}}, \boldsymbol{p}) \text{ independent of } L_{\mathcal{X}} \quad & (2.51)\\
\Leftrightarrow\ \exists\ \phi(w) \text{ such that} & & \theta(l,w) \triangleq \tfrac{f(l+1)w-\phi(w)}{f(l)} \text{ independent of } l \quad & (2.52)\\
\Leftrightarrow\ \exists\ \phi(w) \text{ such that} & & \tfrac{\partial\theta}{\partial l} = \tfrac{f(l)f'(l+1)w-[f(l+1)w-\phi(w)]f'(l)}{f^2(l)} = 0 \quad & (2.53)\\
\Leftrightarrow & & \tfrac{\phi(w)}{w} = f(l+1) - \tfrac{f(l)f'(l+1)}{f'(l)} \text{ independent of } l \quad & (2.54)\\
\Leftrightarrow & & \tfrac{f''(l)}{f'(l)} = \tfrac{f''(l+1)}{f'(l+1)} \quad & (2.55)\\
\Leftrightarrow & & \tfrac{d}{dl}\ln\tfrac{d}{dl}f(l) = 0 \quad & (2.56)\\
\Leftrightarrow\ \exists\ r,s,c \in \mathbb{R} \text{ such that} & & f(l) = r\int_1^l e^{sx}dx + c. \quad & (2.57)
\end{aligned}$$

Recall that $\phi(w)$ models the merge step of Huffman(-like) coding; it is of thus of the form $\phi(w_{n-1} + w_n) = a \cdot (w_{n-1} + w_n)$ in (2.28). The form of (2.28) yields (2.51) $\Leftrightarrow$ (2.52). Equation (2.53) is obtained due to independence iff zero partial derivative, as is (2.55); in both cases, we know $f'(l) > 0$. Solving this, we find that the proposed technique works for and only for linear and exponential functions. This was noted without formal proof in [45].

In addition to being Huffman codable, these penalties have another distinguishing characteristic:

Definition 10 *When referring to averages,* additivity *on function f is the following property for any product distribution $\boldsymbol{p} \times \boldsymbol{q}$ and lengths of the form k_i and l_j: $f^{-1}\{\sum_i \sum_j p_i q_j f(k_i + l_j)\} = f^{-1}\{\sum_i p_i f(k_i)\} + f^{-1}\{\sum_j q_j f(l_i)\}$.*

We see how this is a desirable property in that the matter of whether an outcome is considered as one message or two is not of arbitrary importance as it would be for other penalties. Two independent messages have the same summed penalty as the penalty on the product distribution for the code formed by concatenating the messages. Optimizing for combined items can make the one bit range of the inequality of (2.21) an arbitrarily small proportion of the overall penalty, one of the fundamental properties of Shannon source coding.

The exponential and linear means of the form $G_b(L_{\mathcal{X}}, \boldsymbol{p})$ are the only quasilinear average codeword length penalties for binary codes [5]. Knuth in [53] noted the connection between this and the limited ability to generalize Huffman codes. As additivity is a desirable property, however, this family of problems solved is nonetheless significant.

Still, there are motivations for extending the penalties to a different family, especially when only one message is encoded in a given time period and/or when there is a (nondeterministic) stochastic model of encoding frequency. This may be done via algorithms unrelated to the Huffman algorithm [24,58,59]; we consider variants of one such algorithm in Chapter 4. There are also alternative methods for the extension of Huffman coding to other penalties. The best-known methods involve generalization of arithmetic operations, e.g., using maximization or ordered pairs [18, 39, 53]. The method we consider next, however, does not.

Chapter 3

Redundancy penalties

In this chapter, we introduce a new penalty class, finding this new framework to shed new light on previously proposed problems, including improved solutions to one problem in particular, and a single expression for a set of source coding bounds.

3.1 Motivation, method, and properties

3.1.1 d-average b-redundancy (DABR)

An additional logical penalty might be one that minimizes some measure of "deviation" of actual codeword lengths from their ideal values.

Definition 11 Pointwise b-redundancy, *$r_b(i)$, is the difference between actual and ideal codeword lengths, that is, $r_b(i) \triangleq l_i - l_i^\dagger(b, \boldsymbol{p})$, where previously defined $l_i^\dagger(b, \boldsymbol{p})$ is a function of b and $\boldsymbol{p}$, with the explicit value given in (2.14):*

$$l_i^\dagger(b, \boldsymbol{p}) = -\frac{1}{1+b} \lg p_i + \lg \sum_j p_j{}^{\frac{1}{1+b}} \tag{3.1}$$

for $-1 < b < +\infty$. We define $l_i^\dagger$ via limiting for $b = -1$ and $b = +\infty$. For $b = -1$, if $M = \{m_i \mid p_1 = p_i\}$, $l_i^\dagger(-1, \boldsymbol{p})$ is equal to $\lg |M|$ for M and $+\infty$ otherwise. For $b = +\infty$, $l_i^\dagger(+\infty, \boldsymbol{p}) = \lg |\mathcal{X}| \; \forall \; i \in \mathcal{X}$. (We do not consider the case $b < -1$; in this *case $l_i^\dagger$ should be 0 for l_1 and $+\infty$ for all other l_i, but we ignore this, viewing it as a degenerate case.)*

We should stress that pointwise b-redundancy applies to an individual codeword with a given probability, measuring how the codeword's length varies from the ideal codeword length for exponential Huffman (minimal β-average) coding. It is easily seen that pointwise b-redundancy may be negative for a given codeword and must be negative for at least one codeword in any code for which the Kraft inequality holds with equality.

Because b-redundancy may be negative for a given codeword, the moment about zero, $\{E_{\boldsymbol{p}}[(r_b(X))^d]\}^{1/d}$, cannot be used as a penalty. (Take, for example, $\boldsymbol{p} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $b = 0$ and $d = 5$. In this case, $\{E_{\boldsymbol{p}}[(r_b(X))^d]\}^{1/d}$ is not real or unique, being the root of a negative number.) The analogue of moment for random variables that take on negative values is $\frac{1}{d} \lg E_{\boldsymbol{p}}[2^{dX}]$ [94], also known as the β-average [7], a function we have seen before.

Applying this yields a new penalty:

Definition 12 d-average b-redundancy*, or* DABR*, is*

$$R^{b,d}(L_{\mathcal{X}}) \triangleq \frac{1}{d} \lg E_{\boldsymbol{p}}[2^{dr_b(X)}] = \frac{1}{d} \lg \sum_i p_i 2^{d[l_i - l_i^\dagger(b,\boldsymbol{p})]}. \tag{3.2}$$

The goal now is to minimize DABR, that is, to solve

$$L_{\mathcal{X}}^{b,d} \triangleq \mathop{\arg\min}_{\substack{\sum_i 2^{-l_i} \leq 1, \\ l_i \in \mathbb{N}}} R^{b,d}(L_{\mathcal{X}}) = \mathop{\arg\min}_{\substack{\sum_i 2^{-l_i} \leq 1, \\ l_i \in \mathbb{N}}} \frac{1}{d} \lg E_{\boldsymbol{p}}\{2^{d[r_b(X)]}\} \tag{3.3}$$

where the Kraft inequality restriction on l_i will be implicit from here on.

Note that since each $l_i^\dagger$ is a function of $\boldsymbol{p}$, DABR is not linear with $\boldsymbol{p}$. Thus, it is a subset of a problem more general than that given in (2.2) and (2.3), and thus the analysis of Section 2.6 no longer applies. The more general problem formulation, previously introduced (2.1), is as follows:

$$\begin{array}{ll}
\text{Given} & \boldsymbol{p} \in \Gamma^0 \\
 & f : \mathbb{N} \times [0,1] \to \mathbb{R} \cup \{+\infty\} \\
\text{Minimize}_{\{L_{\mathcal{X}}\}} & \sum_i f(l_i, p_i) \\
\text{subject to} & \sum_i 2^{-l_i} \leq 1 \\
 & l_i \in \mathbb{N}
\end{array} \qquad (3.4)$$

(In DABR, $f(l_i, p_i)$ should take on negative values for $d < 0$.) We later concern ourselves with other subproblems of forms (2.1), (2.2), and (2.3), but, because we will find that minimal DABR coding is a natural extension of Huffman coding and exponential Huffman coding, we first examine the DABR penalty. Below we find that it actually covers a number of familiar cases.

For $b \in (-1, +\infty)$ and $d \in \mathbb{R}$, we may substitute for $r_b(i)$ as follows:

$$\begin{aligned}
R^{b,d}(L_{\mathcal{X}}) &= \frac{1}{d} \lg E_{\boldsymbol{p}}[2^{d r_b(X)}] & (3.5)\\
&= \frac{1}{d} \lg \frac{1}{\sum_j p_j^{\frac{1}{1+b}}} E_{\boldsymbol{p}}[p(X)^{\frac{d}{1+b}} 2^{dl(X)}] & (3.6)\\
&= \frac{1}{d} \lg \sum_i \left[\frac{p_i^{\frac{1+b+d}{1+b}}}{\sum_j p_j^{\frac{1}{1+b}}} 2^{dl_i} \right]. & (3.7)
\end{aligned}$$

Since the summation in the denominator is strictly positive and constant with different length ensembles we may remove it

$$R'^{b,d}(L_{\mathcal{X}}) \triangleq \frac{1}{d} \lg \sum_i p_i^{\frac{1+b+d}{1+b}} 2^{dl_i} \qquad (3.8)$$

or normalize

$$R_0^{b,d}(L_{\mathcal{X}}) \triangleq \frac{1}{d} \lg \sum_i \left[\frac{p_i^{\frac{1+b+d}{1+b}}}{\sum_j p_j^{\frac{1+b+d}{1+b}}} 2^{dl_i} \right], \qquad (3.9)$$

which reduces the problem to an exponential Huffman coding problem. Thus the linear time algorithms and general properties derived for the problem solved by exponential Huffman coding extend to this two-dimensional version.

Definition 13 DABR Huffman coding *is the application of exponential Huffman coding for exponent d and modified weights $w_i^{b,d} \triangleq p_i^{\frac{1+b+d}{1+b}}$ (or the normalized forms in (3.9)). This is feasible when dealing with $b \in (-1, +\infty)$ and $d \in \mathbb{R}$.*

This algorithm allows for the possibility of ties. As with the exponential case, a bottom-merge version of DABR Huffman coding deterministically returns the minimum optimal code. This is easily seen by reduction to the exponential case with modified weights $w_i^{b,d}$.

Unlike with exponential Huffman coding, however, optimal l_i may decrease with i. This occurs for $b + d + 1 \leq 0$. For limit values of b and d, we take the limit of b first, so we derive the proper $l_i^{\dagger}$'s, then d. For all $\boldsymbol{p}$, if $d < -1$, the optimal code is not a function of $\boldsymbol{p}$; it is always a unary code. (For $b > -d - 1$, the most probable item has the shortest codeword; for $b < -d - 1$, the least probable item has the shortest codeword; and for $b = -d - 1$, the codeword assignments do not matter.)

Thus the range of nontrivial cases for minimal DABR codes for a given probability space may be considered to be parameterized by $b \times d \in [-1, +\infty] \times [-1, +\infty]$, as shown in Figure 3.1. The crosshatched region is the one in which the least probable item has the shortest codeword, and the rest of the shaded region contains the other nonconvex penalty points.

Equation (3.9) puts the problem into a form that is a superset of (2.3) and a subset of (3.4), known as quasilinear with a weight function:

$$\begin{array}{ll} \text{Given} & \boldsymbol{p} \in \Gamma^0 \\ & \text{strictly monotonic } f \in \mathcal{C}^0 : \mathbb{R}^+ \to \mathbb{R}^+ \\ & \varphi \in \mathcal{C}^0 : [0,1] \to \mathbb{R}^+ \\ \text{Minimize}_{\{L_\mathcal{X}\}} & f^{-1}\left(\sum_i \varphi(p_i) f(l_i) / \sum_i \varphi(p_i)\right) \\ \text{subject to} & \sum_i 2^{-l_i} \leq 1 \\ & l_i \in \mathbb{N} \end{array} \tag{3.10}$$

Instances of this problem can, for $|\mathcal{X}| < +\infty$, always be normalized and thus reduced to a quasilinear problem without weights.

3.1.2 Special cases

The problem of d-average b-redundancy actually includes a number of previously examined problems, each of which is a subset of the $b \times d$ quadrant. If $b = +\infty$, the modified problem is equivalent to the problem solved via the exponential Huffman coding algorithm, as

$$\frac{1}{d} \lg E_{\boldsymbol{p}}\{2^{d[r_b(X)]}\} = \frac{1}{d} \lg \sum_i p_i \left[\frac{2^{dl_i}}{2^{dl_i^\dagger(b,\boldsymbol{p})}}\right] = \frac{1}{d} \lg E_{\boldsymbol{p}}[2^{dl(X)}] - \lg |\mathcal{X}|, \tag{3.11}$$

so minimizing this minimizes $\frac{1}{d} \lg E_{\boldsymbol{p}}[2^{dl(X)}]$.

If $b = 0$, the actual codeword lengths are compared to the Shannon ideal codeword length of $-\lg p_i$, sometimes called *self-information* [50]. The following definition is useful:

Definition 14 Pointwise redundancy*, or pointwise* 0*-redundancy, is* $r_0(i) = l_i - l_i^\dagger(0, \boldsymbol{p}) = l_i - (-\lg p_i) = l_i + \lg p_i$*, the difference between the actual codeword length and the Shannon ideal codeword length for codeword* c_i*.*

Unless it is uniformly 0, pointwise redundancy must be negative for at least one codeword of the optimal code, but for any uniquely decodable code the average pointwise redundancy — or *average redundancy* for short — is never negative. Furthermore, since the β-average function is nondecreasing, $R_{\boldsymbol{p}}^{0,d}(L_{\mathcal{X}}) \geq 0 \; \forall \; d \geq 0$, including $R_{\boldsymbol{p}}^{0,+\infty}(L_{\mathcal{X}})$, maximal redundancy.

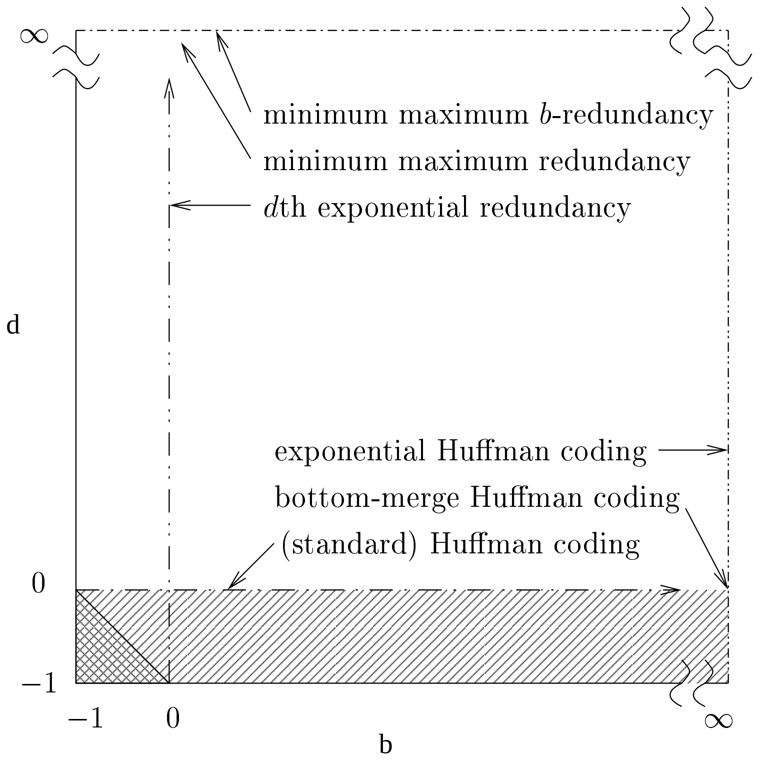

∞
minimum maximum b-redundancy
minimum maximum redundancy
dth exponential redundancy
d
exponential Huffman coding
bottom-merge Huffman coding
(standard) Huffman coding
0
−1
−1
0
∞
b

Figure 3.1: Parameter space for minimal DABR coding with notable special cases

Definition 15 *d*th exponential redundancy *is DABR for* $b = 0$*:*

$$\begin{aligned} L_{\mathcal{X}}^{0,d} &= \arg\min_{L_{\mathcal{X}}} R_{\boldsymbol{p}}^{0,d}(L_{\mathcal{X}}) && (3.12) \\ &= \arg\min_{L_{\mathcal{X}}} \frac{1}{d} \lg E_{\boldsymbol{p}}\{2^{d[l(X)+\lg p_X]}\} && (3.13) \\ &= \arg\min_{L_{\mathcal{X}}} \frac{1}{d} \lg \sum_i \left[\frac{p_i^{1+d}}{\sum_j p_j^{1+d}} 2^{dl_i} \right]. && (3.14) \end{aligned}$$

We comment further on this problem in the next section.

An alternative formulation for the Huffman coding problem is the minimization of average redundancy, which may be called the 0th exponential redundancy problem. Note that, in fact, if we have $d \to 0$ for any value of b in equation (3.3) then

$$\lim_{d\to 0}\frac{1}{d}\lg E_{\boldsymbol{p}}\{2^{d[r_b(X)]}\} = E_{\boldsymbol{p}}[r_b(X)] = E_{\boldsymbol{p}}[l(X)] - \sum_i p_i l_i^\dagger(b,\boldsymbol{p}) \tag{3.15}$$

and so this is equivalent to the Huffman coding problem for any value of b. For $b = +\infty$, approaching $d \uparrow 0$ or $d \downarrow 0$, as discussed, is equivalent to top-merge or bottom-merge Huffman coding, respectively.

Because taking $d \to 0$ always yields Huffman coding, DABR does not include as a special case minimization of $\sum_i p_i^\gamma l_i$ for $\gamma \neq 1$. Clearly, however, this may be solved using Huffman coding on modified weights ($w_i = p_i^\gamma$). In the context of minimal DABR coding, one can use the one-dimensional subset defined by the line $b = d/(\gamma-1) - 1$ to view the solution of minimizing $\sum_i p_i^\gamma l_i$ as that lying in an open interval on this line with one endpoint at $d = 0$ ($b = -1$).

3.1.3 Bounds

One can easily see that if we relax the integer constraint on length for minimizing d-average b-redundancy, the real-valued solution is not $L_{\mathcal{X}}^\dagger$, but some different $L_{\mathcal{X}}^\ddagger$. By substituting the solution in (3.9), we find

$$\begin{aligned} l_i^\ddagger &= -\frac{1+b+d}{(1+b)(1+d)}\lg p_i + \lg\left(\sum_j p_j^{\frac{1+b+d}{(1+b)(1+d)}}\right) && (3.16)\\ &= -\omega \lg p_i + \lg \sum_j p_j{}^\omega && (3.17)\end{aligned}$$

where $\omega \triangleq \frac{1+b+d}{(1+b)(1+d)} = 1 - \frac{bd}{(1+b)(1+d)}$.

Note that when the values of b and d are exchanged, the ideal solutions remain the same. This problem thus has a high degree of symmetry. However, because the problem itself is not symmetric, the symmetry of integer solutions is not perfect, as we can see in Figure 3.2. The dashed curves indicate transitions between regions of optimality for $\boldsymbol{p} = (0.58, 0.12, 0.11, 0.1, 0.09)$, regions marked (1) $L_{\mathcal{X}} = \{1,2,3,4,4\}$, (2) $L_{\mathcal{X}} = \{1,3,3,3,3\}$, (3) $L_{\mathcal{X}} = \{2,2,2,3,3\}$, (4) $L_{\mathcal{X}} = \{4,4,3,2,1\}$, and (5) $L_{\mathcal{X}} = \{3,3,2,2,2\}$. The dotted lines indicate the ($+\infty$) asymptotic behavior of the limits

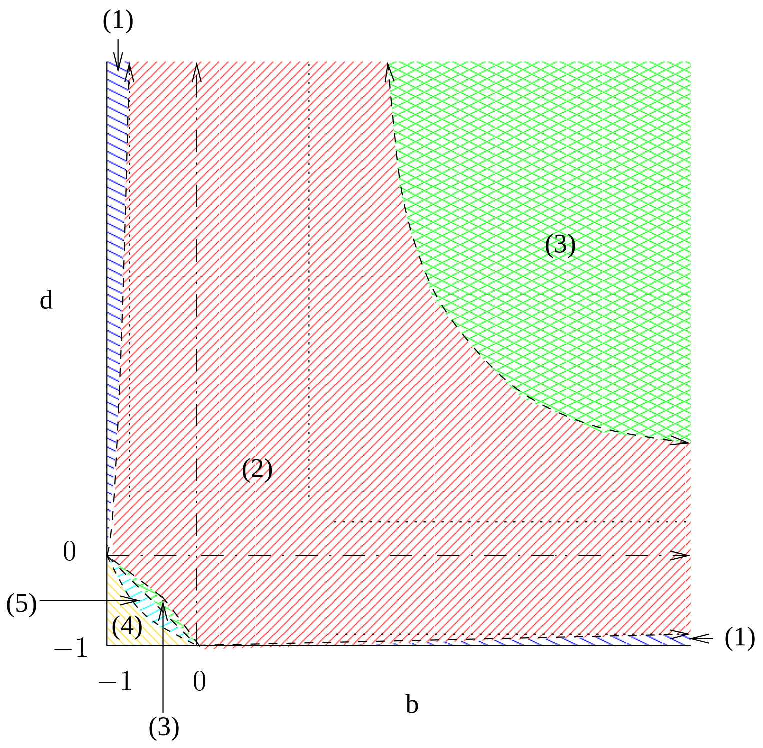


Figure 3.2: Regions of optimality in parameter space for minimal DABR coding, $\boldsymbol{p} = (0.58, 0.12, 0.11, 0.1, 0.09)$

between regions. Note that $b+d+1=0$ divides nonincreasing length ensembles from nondecreasing.

For any $b = 0$ or $d = 0$, $\omega = 1$, and the ideal length is the same as that of linear coding, $-\lg p_i$. This should be expected due to the Huffman coding line ($d = 0$) and symmetry. A similar result, noted by Nath, requires the following definition:

Definition 16 *A penalty that is quasilinear with a weight function — as in equation (3.10) — is* translative *if the following property holds for functions f and φ, distribution $\boldsymbol{p}$, and lengths of the form l_i: $f^{-1}\{\sum_i \varphi(p_i) f(l_i+1)\} = 1 + f^{-1}\{\sum_i \varphi(p_i) f(l_i)\}$.*

Nath found that the penalty class we call dth exponential redundancy — those of DABR form with $b = 0$ — contain the only translative penalties that are quasilinear with a weight function and for which the real-valued and integer-valued solutions corresponded if and only if the input was dyadic.

Using the above equation and the Shannon code analogue, $\lceil l_i^\ddagger \rceil$, we can find bounds for the optimal DABR when $b \geq -1$, $d \geq -1$ and $b + d \geq -1$:

$$\begin{aligned} \alpha b H_\omega(\boldsymbol{p}) &\leq R'^{b,d}(L^*_{\mathcal{X}}) < \alpha b H_\omega(\boldsymbol{p}) + 1 && (3.18) \\ \alpha b(H_\omega(\boldsymbol{p}) - H_\alpha(\boldsymbol{p})) &\leq R^{b,d}(L^*_{\mathcal{X}}) < \alpha b(H_\omega(\boldsymbol{p}) - H_\alpha(\boldsymbol{p})) + 1 && (3.19) \\ \alpha(bH_\omega(\boldsymbol{p}) + H_{1+\alpha d}(\boldsymbol{p})) &\leq R_0^{b,d}(L^*_{\mathcal{X}}) < \alpha(bH_\omega(\boldsymbol{p}) + H_{1+\alpha d}(\boldsymbol{p})) + 1 && (3.20) \end{aligned}$$

where we recall $\alpha = \frac{1}{1+b}$ and $\omega = \frac{1+b+d}{(1+b)(1+d)}$. As before, equality holds iff the ideal solution $L^\ddagger_{\mathcal{X}}$ has all integer lengths. For $b = +\infty$, (3.18) reduces to (2.21) — with the linear (Shannon) case as a special case — and for $b = 0$, (3.20) reduces to

$$H_{1+d}(\boldsymbol{p}) \leq R_0^{0,d}(L^*_{\mathcal{X}}) < H_{1+d}(\boldsymbol{p}) + 1, \quad (3.21)$$

a result of Nath in [76], which may also be stated as

$$0 \leq R'^{0,d}(L^*_{\mathcal{X}}) < 1. \quad (3.22)$$

Equation (3.22) can be restated in terms of a function proposed by Rényi, one he called the gain of information of order α but that we call the Rényi divergence. If we substitute $\gamma = 1 + d$ and $\boldsymbol{q} = 2^{-L_{\mathcal{X}}}$ (the probability vector for which the ith probability is 2^{-l_i}), then

$$\begin{aligned} D_\gamma(\boldsymbol{p} \parallel \boldsymbol{q}) &\triangleq \frac{1}{\gamma-1} \lg \left(\sum_k \frac{p_k^\gamma}{q_k^{\gamma-1}} \right) \\ &= \frac{1}{d} \lg \sum_i p_i^{1+d} 2^{dl_i} && (3.23) \\ &= R'^{0,d}(L_{\mathcal{X}}) \end{aligned}$$

so $0 \leq D_{1+d}(\boldsymbol{p} \parallel 2^{-L^*_{\mathcal{X}}}) < 1$. This is not surprising given the relationship between divergence and Huffman coding noted by Longo and Galasso in [65]. As with the previous cases, equality holds iff the ideal solution $L^\ddagger_{\mathcal{X}}$ has all integer lengths.

There are yet more ways of viewing these redundancy, length, and entropy properties that have desirable analytic properties the above do not [5, 23]. However, such measurements have bijective mappings with the above and do not alter the optimal solution space. Thus we do not discuss them here.

3.2 Minimizing maximal pointwise redundancy

Standard Huffman coding minimizes average redundancy, as illustrated in (3.15); Huffman's original paper [43] is in fact titled "A Method for the Construction of Minimum-Redundancy Codes." As average pointwise (0-)redundancy has been well-understood for some time, Drmota and Szpankowski decided to explore the previously overlooked minimization of maximal pointwise redundancy in [24]. (This, although referred to as "minimax redundancy," is different from the "minimax redundancy" considered for unknown probability vectors, a problem we do not consider here.)

Note that the minimax redundancy problem is equivalent to minimizing dth exponential redundancy as $d \to +\infty$:

$$\begin{aligned}
\lim_{d\to+\infty} \frac{1}{d} \lg E_{\boldsymbol{p}}[2^{dr_0(X)}] &= \lim_{d\to+\infty} \left[r_{max} + \frac{1}{d} \lg P_X\{r_0(X) = r_{max}\} + O\left(\frac{1}{d2^d}\right)\right] && (3.24)\\
&= r_{max} && (3.25)
\end{aligned}$$

where $r_{max} \triangleq \max_i r_0(i)$. Thus, considering $d \in [0, +\infty]$, dth exponential redundancy is a subproblem with a parameter that varies solution values between minimizing average redundancy (Huffman coding) and minimizing maximum redundancy; such a range of problems and solutions was sought by Drmota and Szpankowski. This range was previously derived axiomatically by Nath as noted in Section 3.1.3. The version of the minimal DABR coding solution applying to this subproblem was found shortly thereafter in [78], although not generalized to $b \neq 0$, as with the previous section, or to $d = +\infty$, the focus of this section.

In this section, we first summarize the algorithm proposed in [24] to find a (nonunique) code with minimum maximum pointwise redundancy. We then find an improvement based on minimal DABR Huffman coding by finding a value $d_{mm} \in \mathbb{R}$ such that (3.13) is optimized for all $d \geq d_{mm}$, and thus for $d \to +\infty$. The shaded region in Figure 3.3 illustrates the range of d values which correspond to problems that

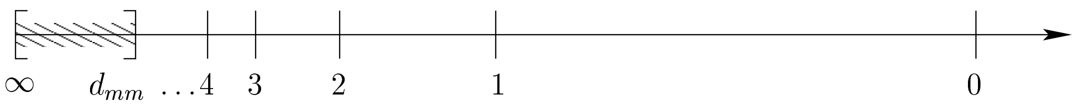


Figure 3.3: Range of d — ordered by $\frac{1}{1+d}$ — with $[d_{mm}, +\infty]$ shaded

such a solution would be guaranteed to minimize. Finally, we will propose alternative algorithms to find minimax redundancy codes, algorithms that are more efficient than either the Drmota and Szpankowski algorithm or the algorithm that finds d_{mm}.

3.2.1 Minimax redundancy as a generalized Shannon code

Definition 17 *A* generalized Shannon code *is a prefix-free code for which,* $\forall\ i \in \mathcal{X}$, $\lfloor l_i^\dagger(0,\boldsymbol{p})\rfloor \leq l_i \leq \lceil l_i^\dagger(0,\boldsymbol{p})\rceil$. *(Recall* $l_i^\dagger(0,\boldsymbol{p}) = -\lg p_i$.*)*

Let $\langle x\rangle$ denote the fractional part of x, i.e., $\langle x\rangle \triangleq x - \lfloor x\rfloor$. Now let $\{t_i\}$ be a permutation of $\{1,2,\ldots,n\}$ such that $\forall\ i<j, \langle l_{t_i}^\dagger(0,\boldsymbol{p})\rangle \leq \langle l_{t_j}^\dagger(0,\boldsymbol{p})\rangle$ $(n = |\mathcal{X}|)$. Let j^* be the maximal j such that

$$\sum_{i=1}^{t_{j-1}} p_i 2^{\langle l_{t_i}^\dagger(0,\boldsymbol{p})\rangle} + \frac{1}{2}\sum_{i=t_j}^{n} p_i 2^{\langle l_{t_i}^\dagger(0,\boldsymbol{p})\rangle} \leq 1, \tag{3.26}$$

that is, the Kraft inequality holds for the following generalized Shannon length ensemble:

$$l_i = \begin{cases} \lfloor l_{t_j}^\dagger(0,\boldsymbol{p})\rfloor, & j < j^* \\ \lceil l_{t_j}^\dagger(0,\boldsymbol{p})\rceil, & j \geq j^* \end{cases} \tag{3.27}$$

Then $\forall\ L_\mathcal{X},\ R_{\boldsymbol{p}}^{0,+\infty}(L_\mathcal{X}) \geq \langle 1 - l_{t_{j^*}}^\dagger(0,\boldsymbol{p})\rangle$, i.e., the code presented in (3.27) achieves minimax redundancy. A proof and a practical $O(n \log n)$ algorithm may be found in [24].

This code, however, is in no sense unique in minimizing maximal redundancy. It may be one of many optimal codes, as decrementing a value of l_i may not necessarily violate the Kraft inequality. In addition, if there is more than one codeword with $\langle l_t^\dagger(0,\boldsymbol{p})\rangle = \langle l_{t_{j^*}}^\dagger(0,\boldsymbol{p})\rangle$, a problem arises; because there is no method specified to break ties, the algorithm does not uniquely determine which of these items have $l_i = \lfloor l_i^\dagger(0, \boldsymbol{p}) \rfloor$ and which have $l_i = \lceil l_i^\dagger(0, \boldsymbol{p}) \rceil$.

It is therefore instructive to first make a minor modification to the algorithm proposed in [24] by forcing codewords with $\langle l_t^\dagger(0, \boldsymbol{p}) \rangle = \langle l_{t_{j^*}}^\dagger(0, \boldsymbol{p}) \rangle$ to be of identical redundancy — if there are no ties, this is the same as the code previously presented. The redundancy at all these points is the maximal possible among optimal codes in order to satisfy both the Kraft inequality and the previous assertion. Minimax redundancy is identical for the result of this modified algorithm, which chooses j^* to be the maximal j such that (3.26) holds and $\langle l_{t_j}^\dagger(0, \boldsymbol{p}) \rangle \neq \langle l_{t_{j-1}}^\dagger(0, \boldsymbol{p}) \rangle$. If the maximum redundancy is r_{max}, then $0 \leq r_{max} < 1$ and $\forall\ i \in \mathcal{X},\ r_0(i) \in (r_{max} - 1, r_{max}]$, a unit interval about 0. The modification thus, in addition to making the algorithm invariant to permutation of input, insures the infimum point in this interval is not a minimum.

Lengthening any codeword increases pointwise redundancy by one and thus results in the value being outside this interval, strictly increasing the maximal redundancy. Thus, in some sense, this generalized Shannon code is the "worst" minimax redundancy code, as no value of l_i is greater in any optimal code than the corresponding one in this code.

3.2.2 Minimax redundancy as minimal DABR

We now present an idealization of how to improve upon the above "worst" code. Due to its complexity, we do not propose it as a practical algorithm, but using it we may find the aforementioned d_{mm} for which the codes with optimal dth exponential redundancy are the same as those with minimax redundancy, the d_{mm} illustrated by Figure 3.3. Such a d_{mm} would reduce minimax redundancy to minimal d_{mm}-average 0-redundancy. The resulting code would not only optimize the maximal redundancy, but lesser-order terms of equation (3.24) as well, resulting in a full tree that could be found with an efficient Huffman-like algorithm. A more efficient algorithm for the identical code is introduced in the next section, but this approach aids understanding of the problem.

First order a queue by the value of $r_0(i) = l_i - l_i^\dagger(0, \boldsymbol{p})$ from the largest to smallest value as follows: Begin with the results of the above modified generalized Shannon construction. Then insert items starting with $m_{t_{j^*}}$ and continuing in order until m_{t_n}, then restarting at m_{t_1} and continuing until $m_{t_{j^*-1}}$.

Denote the items in the queue $\mathcal{Q}$ and let $\bar{\mathcal{Q}} \triangleq \mathcal{X}\backslash\mathcal{Q}$, the set of values known to be optimal, initially the null set. Let q_i denote (the index for) the ith element of the queue, and let n_i denote (the index for) the ith element removed from the queue.

The idealized approach is first to find the expression for the dth exponential redundancy, then to decrement the lengths corresponding to the largest terms that can be decremented. We reorder and then repeat until the Kraft inequality is satisfied and we can make no more improvement. If we can find a value of d_{mm} such that solving (3.13) for any $d \geq d_{mm}$ yields an identical result, we may use minimal DABR Huffman coding as an equivalent method.

Let $\min^+_{i,j} \gamma_{i,j}$ denote the minimum strictly positive value of $\gamma_{i,j}$. Assign $\delta_{\boldsymbol{p}} \triangleq \min^+_{i,j}\langle l_i^\dagger(0,\boldsymbol{p}) - l_j^\dagger(0,\boldsymbol{p})\rangle$. Now let $d_{mm} = \frac{1}{\delta_{\boldsymbol{p}}} \lg \frac{2}{p_n} > 1$. Suppose $d \geq d_{mm}$, and suppose the following is invariant in the algorithm: Each item not in the queue is known to be at the proper value for the optimal $l_i = l_i^*$, and each item in the queue is known to have value $l_i \geq l_i^*$. Note that, since $\bar{\mathcal{Q}} = \emptyset$ initially, this is true after the modified algorithm previously specified, so we need only preserve this invariant property. Let the optimal subset $\mathcal{O} \triangleq \{q_i \mid \langle l_i^\dagger(0,\boldsymbol{p})\rangle = \langle l_1^\dagger(0,\boldsymbol{p})\rangle\} \subseteq \mathcal{Q}$, which, unless there are ties in $\langle l_i^\dagger(0,\boldsymbol{p})\rangle$, should be just $\{q_1\}$. Let $E_{\bar{\mathcal{Q}}} \triangleq \sum_{n_i \in \bar{\mathcal{Q}}} p_i 2^{dr_0(i)}$. Then we have

$$E_{\boldsymbol{p}}[2^{dr_0(X)}] = \sum_{q_o \in \mathcal{O}} p_{q_o} 2^{d[l_{q_o} - l_{q_o}^\dagger(0,\boldsymbol{p})]} + \sum_{q_i \in \mathcal{Q}\backslash\mathcal{O}} p_i 2^{d[l_{q_i} - l_{q_i}^\dagger(0,\boldsymbol{p})]} + E_{\bar{\mathcal{Q}}}. \tag{3.28}$$

Since members of $\bar{\mathcal{Q}}$ are known to have optimal values, the only way in which we can improve upon these codeword lengths is to shorten other values. If it is impossible to shorten the length of certain codewords and still satisfy the Kraft inequality, we may remove them from the queue and retain the invariant assumptions as follows.

We may first choose between decrementing lengths in $\mathcal{O}$ and altering other lengths in the queue, assuming there are any. Either way, we decrease the value in (3.28). Note that the contribution of $\mathcal{O}$ is the first term, $\sum_{q_o \in \mathcal{O}} p_{q_o} 2^{d[l_{q_o} - l_{q_o}^\dagger(0,\boldsymbol{p})]}$, while the contribution of other queue members is the second term, $\sum_{q_i \in \mathcal{Q}\backslash\mathcal{O}} p_i 2^{d[l_{q_i} - l^\dagger_{q_i}(0,\boldsymbol{p})]}$. If we can show that (3.28) is reduced more by decrementing (by 1) any of l_{q_o} (where q_o may be any member of $\mathcal{O} = \{q_1, \ldots, q_{|\mathcal{O}|}\}$) than by eliminating the second term altogether (since it is greater than 0 for any valid solution), we will have shown that it is optimal to decrement values of l_{q_o} before considering values in $\mathcal{Q}\backslash\mathcal{O}$:

$$\begin{aligned}
\sum_{q_i \in \mathcal{Q}\backslash\mathcal{O}} p_i 2^{d[l_{q_i} - l^\dagger_{q_i}(0,\boldsymbol{p})]} &< 2^{d[l_{q_{o^*+1}} - l^\dagger_{q_{o^*+1}}(0,\boldsymbol{p})]} && (3.29)\\
&\le \frac{2^{d[l_{q_o} - l^\dagger_{q_o}(0,\boldsymbol{p})]}}{2^{d\delta_{\boldsymbol{p}}}} \ \forall\ q_o \in \mathcal{O} && (3.30)\\
&\le \frac{p_n}{2} 2^{d[l_{q_o} - l^\dagger_{q_o}(0,\boldsymbol{p})]} \ \forall\ q_o \in \mathcal{O} && (3.31)\\
&< p_n \{2^{d[l_{q_o} - l^\dagger_{q_o}(0,\boldsymbol{p})]} - 2^{d[l_{q_o} - 1 - l^\dagger_{q_o}(0,\boldsymbol{p})]}\} \ \forall\ q_o \in \mathcal{O} && (3.32)\\
&< p_{q_o} 2^{d[l_{q_o} - l^\dagger_{q_o}(0,\boldsymbol{p})]} - p_{q_o} 2^{d[l_{q_o} - 1 - l^\dagger_{q_o}(0,\boldsymbol{p})]} \ \forall\ q_o \in \mathcal{O}. && (3.33)
\end{aligned}$$

Since $l_{q_o} - l^\dagger_{q_o}(0,\boldsymbol{p}) = l_{q_1} - l^\dagger_{q_1}(0,\boldsymbol{p})\ \forall\ o \in \mathcal{O}$, the above inequalities do not change with o. Expectation does not exceed maximum, yielding (3.29). Because $\delta_{\boldsymbol{p}}$ is the minimum difference between values of $l_i - l^\dagger_i(0,\boldsymbol{p})$ and $l_j - l^\dagger_j(0,\boldsymbol{p})$, (3.30) holds. Because $d \ge d_{mm}$, (3.31) holds, $d \ge d_{mm} > 1$ yields (3.32), and $p_n < p_{q_o}$ obtains (3.33).

Thus, if we obey the invariants, it is optimal to decrement the tied first terms in the queue before proceeding. These values remain tied for any value of $d \ge d_{mm}$, so any such algorithm for finite d optimally resolves the ties. After decrementing, if we return the member to the end of the queue, order is preserved, as are the aforementioned invariants. We may thus continue until the queue is empty and the Kraft inequality is satisfied with equality.

The problem and thus result is identical for any $d \ge d_{mm} = \delta_{\boldsymbol{p}}^{-1}(1 - \lg p_n)$. Choose a parameter $\tilde{d} \ge d_{mm}$ for ease of computation, e.g., $\tilde{d} = 2^{\lceil \lg d_{mm} \rceil}$. Thus, if we find an optimal $\tilde{d}$th exponential Huffman code for weights $w_i^{0,\tilde{d}} = p_i^{1+\tilde{d}}$, it is a minimax code that, unlike the generalized Shannon code, satisfies the Kraft inequality with equality. This is easily extended to other values of b (for which $\delta_{\boldsymbol{p}} = \min^+_{i,j} \langle l^\dagger_i(b,\boldsymbol{p}) - l^\dagger_j(b,\boldsymbol{p})\rangle$).

Note however that this algorithm for minimax redundancy is not linear time given sorted p_i's as is the case for finite d. Finding d_{mm} requires finding $\min^+_{i,j}\langle l_i^\dagger(b,\boldsymbol{p}) - l_j^\dagger(b,\boldsymbol{p})\rangle$, which may be done via sorting on $\langle l_i^\dagger(b,\boldsymbol{p})\rangle$ and keeping track of minimal difference. This takes $\Theta(n \log n)$ time. In addition, calculating $w_i^{0,\tilde{d}} = p_i^{1+\tilde{d}}$ may vary with $\tilde{d}$, although, with constant time multiplication, it should be $\Theta(\log \frac{\log p_n^{-1}}{\delta_{\boldsymbol{p}}})$. Thus the total algorithm is $\Theta(n[\log n + \log \frac{1}{\delta_{\boldsymbol{p}}} + \log\log \frac{1}{p_n}])$ in time and linear in space.

Also, in spite of the multiple criteria this algorithm optimizes, it does not, in fact, return a unique solution in every instance. Only a secondary criterion, such as that of a minimal code, could resolve ties. As previously discussed, bottom-merging finds the minimum code among optimal codes. We give an example of an optimal code after presenting a more efficient algorithm.

3.2.3 Linear time methods for minimax redundancy

Although the above is useful in understanding a context for minimax redundancy coding, linear time algorithms may also be developed. By keeping d_{mm} as a variable, we may arrive at an algebraic version of the above algorithm. To do this with a Huffman algorithm, we have to keep track of only the first and second order terms, as ties between these pairs of terms can occur only when all terms are tied, this due to the manner in which the Huffman procedure works. We explain why later; first, we present the algorithm.

The aforementioned first and second order terms are $w_i' \triangleq \lim_{d\to+\infty}[w_i(d)]^{(d^{-1})}$ and $w_i'' \triangleq \lim_{d\to+\infty}[w_i(d)]^{-1} \cdot [w_i']^d$, respectively, where leaf nodes have $w_i(d) = p_i^{\frac{1+b+d}{1+b}}$, as previously stated. One may think of w_i' as representing an invertible function of maximal b-redundancy, $w_i' = \left[\sum_{j=1}^n p_j^{\frac{1}{1+b}}\right]^{-1} \cdot 2^{\max_i r_b(i)}$, where $r_b(i) = l_i - l_i^\dagger(b,\boldsymbol{p})$ uses the depth of item i in its current tree as the value l_i at any given point of the algorithm. Note that only $r_b(i)$ is variable; the denominator term is a result of not normalizing the weights at the start of the algorithm. Similarly, w_i'' represents the probability of maximal b-redundancy $P_X\{r_b(X) = \max_j r_b(j)\}$.

The algorithm does not find a value of d_{mm} such that this is also the solution to dth exponential redundancy (or DABR) coding for $d \in [d_{mm}, +\infty)$, as the previous numeric algorithm does. However, the algebraic algorithm is linear time given sorted weights, faster than either the previously considered algorithm or that given by Drmota and Szpankowski in [24].

To implement this algorithm, we let $w'_i = p_i^{\frac{1}{1+b}}$ and $w''_i = p_i$ for the initial case. In comparing items j and k, we consider them as lexicographically ordered pairs — e.g., $w_j = (w'_j, w''_j)$ — so that $w_j \geq w_k$ if and only if either $w'_j > w'_k$ or if $w'_j = w'_k$ and $w''_j \geq w''_k$, as in [53]. In combining items j and k (where $w_j \geq w_k$ as described), the new item will have $\tilde{w}'_j = 2w'_j = 2 \cdot \max(w'_j, w'_k)$. If $w'_j > w'_k$, then $\tilde{w}''_j = w''_j$. Otherwise, $\tilde{w}''_j = w''_j + w''_k$. That is,

$$\tilde{w}_j = \begin{cases} (2w'_j, w''_j) & \text{if } w'_j > w'_k \\ (2w'_j, w''_j + w''_k) & \text{otherwise} \end{cases} \tag{3.34}$$

The reasons for this are easily seen if we view w_i as the representation of maximal redundancy and probability this maximal redundancy is achieved. Take the maximum and add 1 for the additional bit of the codeword, then, if the redundancies are identical, add their probabilities; otherwise, take the probability of the maximal redundancy.

This combining method is a Huffman algebra, satisfying the properties introduced by Knuth in [53]. The Huffman combining criterion, shown by example in Figure 3.4, results in the same process and yields the same results as for the variable case. The remaining weight, $(\frac{32}{19}, \frac{4}{19})$, indicates a maximum redundancy of $\lg \frac{32}{19}$ and a probability of $\frac{4}{19}$ that this redundancy is achieved.

We now show that ties in the w pairs imply ties in all terms of the expansion presented in equation 3.41, or, equivalently, for all aforementioned $d \in [d_{mm}, +\infty)$.

Theorem 7 *If there is a tie in the above w pairs, there is a tie in all terms of the corresponding d expansion.*

Proof: Consider two tied pairs. Note that, in each,

$$w'_i \geq w''^{\frac{1}{1+b}}_i \tag{3.35}$$

l_i	c_i	m_i	$19 \cdot p_i$	$19 \cdot w_i$
1	1	m_1	8	(8, 8)
3	000	m_2	4	(4, 4)
3	001	m_3	3	(3, 3)
3	010	m_4	2	(2, 2)
3	011	m_5	2	(2, 2)

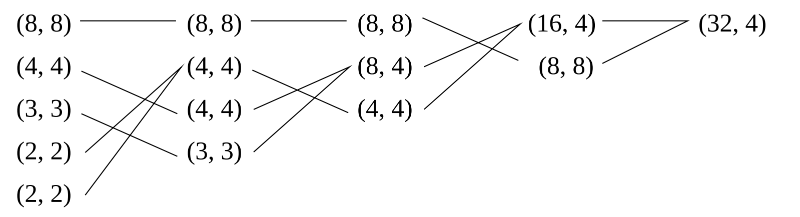


Figure 3.4: Algebraic minimax redundancy coding, $\boldsymbol{p} = \frac{1}{19} \cdot (8, 4, 3, 2, 2)$ (bottom-merge)

l_i	c_i	m_i	$19 \cdot p_i$	$19 \cdot w'_i$
1	1	m_1	8	8
2	01	m_2	4	4
3	001	m_3	3	3
4	0000	m_4	2	2
4	0001	m_5	2	2

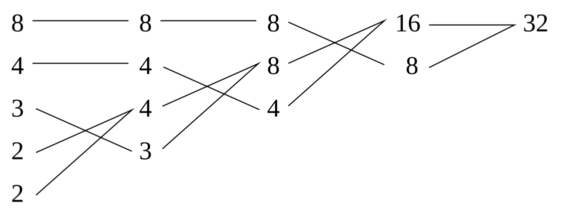


Figure 3.5: Top-merge minimax redundancy coding, $\boldsymbol{p} = \frac{1}{19} \cdot (8, 4, 3, 2, 2)$ (single variable)

because this holds with equality in leaf nodes and the inequality is preserved in the merge step, since $2 \cdot \max(a, b) \geq a + b \geq \max(a, b)$ for $a, b \geq 0$. If inequality (3.35) holds without equality, neither node can be a leaf node, and, due to ordering for the combination step, their four children must be identically weighed. However, this fact can be invoked inductively for either pair of children, also tied, and thus such a tree could not be finite. Therefore, tied pairs arise only in cases for which the inequality holds with equality. Thus, they must be leaf nodes or nodes with two identically weighted children. Inductively, this means the subtrees must be composed of leaf nodes that are *relatively dyadic*, that is, are dyadic when multiplied by a nontrivial common constant. Thus they are equal in all terms, which is what we set out to show. ∎

One can use bottom-merge or top-merge coding to assure that the results are deterministic. If one uses top-merge coding, note that we actually need not keep track of the second variable, as in Figure 3.5, because order in both variables is preserved by favoring combined items.

This variant is actually a special case of a Huffman variation known as the *tree-height measure* problem, not previously considered in the context of minimax redundancy (or DABR) coding. This problem, discussed by Parker in [78], minimizes the maximal value of $v_i + c \cdot l_i$ given $c > 0$ and weight vector $\boldsymbol{v}$. Instead of using $\tilde{v}_j = 2^b(v_j + v_k)$ on the merge step of Huffman coding, the Huffman-like tree-height measure algorithm uses $\tilde{v}_j = c + \max(v_j, v_k)$. For the problem reduction, weights are assigned according to

$$v_i(b) = \frac{1}{1+b} \lg \frac{p_i}{p_n}, \tag{3.36}$$

which is always nonnegative, and $c = 1$. Then this modified Huffman algorithm minimizes

$$\begin{aligned}
\max_i(v_i(b) + c \cdot l_i) &= \max_i \left(l_i + \frac{1}{1+b} \lg \frac{p_i}{p_n} \right) &(3.37)\\
&= \max_i \left(l_i + \frac{1}{1+b} \lg p_i - \lg \left[\sum_j p_j{}^{\frac{1}{1+b}} \right] \right) \\
&\quad + \lg \left[\sum_{j \neq n} p_j{}^{\frac{1}{1+b}} \right] &(3.38)\\
&= \max_i r_b(i) + \lg \sum_{j \neq n} p_j{}^{\frac{1}{1+b}} &(3.39)
\end{aligned}$$

thus minimizing maximal pointwise b-redundancy with a code satisfying the Kraft inequality with equality, also in linear time. The equivalence to the previous algorithm is given by noting $v_i(b) = \lg w_i(b) - \lg w_n(b)$.

Due to ties, if we do not use top-merge coding, this may be one of many possible optimal codes, including codes not optimal for the limit of dth exponential redundancy. For example, consider $\boldsymbol{p} = \left(\frac{8}{19}, \frac{4}{19}, \frac{3}{19}, \frac{2}{19}, \frac{2}{19}\right)$, as in Figures 3.4 and 3.5. For

dth exponential redundancy, $L_{\mathcal{X}} = \{1,2,3,4,4\}$ and $L_{\mathcal{X}} = \{1,3,3,3,3\}$ are both optimal for $d \to +\infty$, optimizing minimax redundancy, then probability of maximal redundancy, then lower-order terms, as

$$\begin{aligned} R_{b,d} &= \tfrac{1}{d} \lg \textstyle\sum_{i\in\mathcal{X}} p_i 2^{d(l_i - l_i^\dagger)} & (3.40) \\ &= \max_i r_b(i) + \\ &\quad \tfrac{1}{d} \lg P_X \left\{ r_b(X) = \max_j r_b(j) \right\} + & (3.41) \\ &\quad O\left(\tfrac{1}{d2^d}\right). \end{aligned}$$

Each term in the expansion has a different asymptotic complexity. As with minimum variance Huffman coding, each additional term further restricts the set of feasible codes to those that satisfy the arg min for the term given the optimization of previous terms. In the above example, all terms are minimized by both the aforementioned sets of lengths. These two codes turn out to be optimal for dth exponential redundancy for all $d > -1$. However, $L_{\mathcal{X}} = \{2,2,2,3,3\}$ also minimizes maximal redundancy and is in fact achieved by the bottom-merge version of the the tree-height measure algorithm. Still, unlike in [24], the output always has a full code tree with all subtrees also optimal and finds a minimizing solution in linear time.

Thus we have linear time methods to find (possibly nonunique) solutions of DABR (equation (3.3)) for all values of $b \times d \in (-1,+\infty] \times (-1,+\infty]$. For $b = -1$ or $d = -1$, we know the unary solution is optimal, and for $b = d = +\infty$, we know the flat solution is optimal, neither case dependent on the actual descending probability values.

The previously explored relations allow similar extensions to alphabetic codes (alphabetic search trees), obtaining optimal codes via modifications of the Hu-Tucker algorithm [39,42]. For nonnegative exponent $d \geq 0$, this is a direct result of [39]. The $d < 0$ case is left as an exercise.

Chapter 4

Generalized quasilinear convex penalties

In this chapter, we generalize an efficient algorithm for finding length-limited codes to an efficient algorithm for finding optimal codes for any penalty of a class we call generalized quasilinear convex penalties. These include the convex quasilinear class proposed by Campbell in [15]. This algorithm may be performed using $O(n^2 \log n)$ time (which in most cases may be reduced to $O(n^2)$ or further), and, in the Campbell case, linear space. Also, for Campbell's problems, we find bounds for optimal codes analogous to those in previous cases.

4.1 Introduction and motivation

Now consider the more general case proposed in equation (2.1):

Definition 18 *Suppose* $f(l_i, p_i) : \mathbb{N} \times [0,1] \to \mathbb{R} \cup \{+\infty\}$ *is monotonically increasing and convex with respect to* $l_i \in \mathbb{N}$*. The* generalized quasilinear convex coding problem *is that of minimizing* $F(L_{\mathcal{X}}, \boldsymbol{p}) = \sum_i f(l_i, p_i)$ *over* $L_{\mathcal{X}}$*, subject to the Kraft inequality and the integer constraint.*

This problem is the one we solve here, thus solving the convex Campbell subproblem case (2.2) as well. Note that the generalized quasilinear problem is in fact quite general, covering problems F_8 through F_{13} in Table 1.1. Heretofore a solution to either this case or Campbell convex case was only known for certain specific instances of these problems. The most general related solution is the one explored in Section 2.5, which uses F merely as a secondary criterion, to break ties among multiple optimal solutions in the linear (or exponential) case. This technique uses bottom-merge Huffman coding, first explored in [84].

Coding for F as a primary criterion has not previously been solved in much generality. The specific Campbell case of $f(l_i) = \alpha l_i + \beta l_i^2$, for given $\alpha, \beta \geq 0$ had been previously considered by Larmore in [58], which presented an $O(n^3)$ time and space algorithm. A version of the algorithm we present can be applied to improve this result to $O(n^2)$ time and linear space.

It is also worthwhile to note that this result of [58] is used in order to construct a polynomial time algorithm for finding a code minimizing a complex nonquasilinear function corresponding to the expected waiting time between information request and receipt, the problem formulated as equation F_{14} of Table 1.1. Our results thus improve this algorithm. In fact, they may be used to find efficient polynomial time algorithms to minimize any of a wide variety of functions, based on the Convex Hull theorem of [58]. We touch upon this in Section 4.8.

We assume, without loss of generality, that the domain of f may be extended to $\{\mathbb{N} \cup \{0\}, [0,1]\}$ and that $f(0, p_i) = 0$. If this is not the case, we may replace f with

$$\tilde{f}(l_i, p_i) = \begin{cases} f(l_i, p_i) - 2f(1, p_i) + f(2, p_i), & l_i > 0 \\ f(l_i, p_i) = 0, & l_i = 0 \end{cases} \tag{4.1}$$

retaining convexity and monotonicity.

4.2 Bounds

Before attempting to solve this problem, let us find bounds for the optimal solution in the quasilinear case, not necessarily assuming convexity. Note that, as in (2.17–2.22), if we extend the domain of $f(l,p)$ to a monotonically increasing function on $\{\mathbb{R}^+,[0,1]\}$, we may bound the solution value as follows. Because the optimal solution to the problem without the integer constraint must have at most the same minimum value, this optimal solution provides a lower bound. We again denote the ith parameter of this solution $l_i^\dagger$. The Shannon-like code $l_i = \lceil l_i^\dagger \rceil$ is achievable and thus yields an upper bound. Note further that the above upper bound is strictly less than $\sum_i f(l_i^\dagger+1,p_i)$ due to f monotonically increasing in the first variable. Thus

$$\sum_i f(l_i^\dagger,p_i) \leq \sum_i f(l_i^*,p_i) \leq \sum_i f(\lceil l_i^\dagger \rceil,p_i) < \sum_i f(l_i^\dagger+1,p_i) \tag{4.2}$$

where each l_i^*, as before, corresponds to the length of the ith codeword in an optimal code. These bounds are analogous to Campbell's interpretation of Rényi entropy and may be useful if the real-valued problem can be solved analytically.

Continuous quasilinear problems (2.3) allow one to define f-entropy via the left-hand side of (4.2). Campbell in [15] defined:

$$H(\boldsymbol{p},f) \triangleq \inf_{\substack{\sum_i 2^{-l_i}\leq 1,\\ l_i\in\mathbb{R}}} f^{-1}\left[\sum_i p_i f(l_i)\right]. \tag{4.3}$$

This is how he derived Rényi entropy, as in equation (2.17).

The translative property noted in Section 3.1.3 is a specific case of the following property for general penalty $F:\mathbb{R}^n\times\Gamma_n^0\to\mathbb{R}$, described by Aczél [6], among others:

Definition 19 *$F(L_{\mathcal{X}},\boldsymbol{p})$ is* translatory *if, for all $c\in\mathbb{R}^+$, $F(L_{\mathcal{X}}+c,\boldsymbol{p}) = F(L_{\mathcal{X}},\boldsymbol{p})+c$, where $L_{\mathcal{X}}+c$ denotes adding c to each item l_i in $L_{\mathcal{X}}$.*

If F is increasing with each l_i, then there exists F-entropy:

$$H(\boldsymbol{p},F) \triangleq \inf_{\substack{\sum_i 2^{-l_i}\leq 1,\\ l_i\in\mathbb{R}}} F(L_{\mathcal{X}},\boldsymbol{p}), \tag{4.4}$$

and, for minimizing input $L^*_{\mathcal{X}}$, if F is translatory, then, due to equation (4.2):

$$H(\boldsymbol{p}, F) \leq F(L^*_{\mathcal{X}}, \boldsymbol{p}) < H(\boldsymbol{p}, F) + 1. \tag{4.5}$$

We may broaden the collection of penalty functions satisfying this penalty by weakening the condition of Definition 19. Replacing the equality with an inequality, we introduce the concept of a *subtranslatory* penalty:

Definition 20 *If $F(L_{\mathcal{X}} + c, \boldsymbol{p}) \leq F(L_{\mathcal{X}}, \boldsymbol{p}) + c$ for all $c \in \mathbb{R}^+$, $F(L_{\mathcal{X}}, \boldsymbol{p})$ may be said to be* subtranslatory*.*

Equation (4.5) still holds for subtranslatory penalties.

If F is continuous quasilinear, as Campbell considered, and if the associated f obeys certain regularity conditions, then we can introduce a necessary and sufficient condition for F to be subtranslatory.

Consider the invertible function $f : \mathbb{R}^+ \to \mathbb{R}^+$, and assume it is real analytic over a relevant compact interval. We may choose this interval to be, for example, $C = [\delta, 1/\delta]$ for some $\delta > 0$. We may take $\delta \to 0$ to show the following argument to be valid over all $\mathbb{R}^+$. We assume f^{-1} is also real analytic (with respect to interval $f(C)$). Thus all derivatives of the function and its inverse are bounded. Without loss of generality, we also assume $f' = \frac{df}{dx}$ is positive; if it were negative, we could replace $f(x)$ with $\tilde{f}(x) = f_{max} - f(x)$ where $f_{max} = \max_{x \in C} f(x) < +\infty$, and the same penalty F would result.

Suppose $\sum_i p_i f'(l_i) \leq f'\left[f^{-1}(\sum_i p_i f(l_i))\right]$. We wish to show this to be equivalent to being subtranslatory. Note that the right-hand side of the inequality may also be written $f'\left[F(L_{\mathcal{X}}, \boldsymbol{p})\right]$, so this may be stated, "The average derivative at the codeword length values is at most the derivative of f at the value of the penalty function for those length values." The inequality is equivalent to $\left[\sum_i p_i f'(l_i)\right] \cdot (f^{-1})'\left[\sum_i p_i f(l_i)\right] \leq 1$. The linear and exponential penalties satisfy the inequality with equality. Also, moment functions $(\sum_i p_i l_i^a)^{1/a}$ satisfy the inequality for $a \geq 1$, since $\left[\sum_i p_i l_i^{a-1}\right]^{\frac{1}{a-1}} \leq \left[\sum_i p_i l_i^a\right]^{\frac{1}{a}}$ leads to $a \cdot \left[\sum_i p_i l_i^{a-1}\right] \leq a \cdot \left[\sum_i p_i l_i^{a-1}\right]^{\frac{a-1}{a}}$, which is $\sum_i p_i f'(l_i) \leq f'\left[f^{-1}(\sum_i p_i f(l_i))\right]$, the inequality we wanted.

Theorem 8 *Given real analytic f and f^{-1},*

$$\sum_i p_i f'(l_i) \leq f' \left\{ f^{-1} \left[\sum_i p_i f(l_i) \right] \right\} \tag{4.6}$$

if and only if penalty F is subtranslatory.

Proof: Let $\epsilon > 0$:

$$\begin{aligned}
& f^{-1}\left[\sum_i p_i f(l_i)\right] + \epsilon & (4.7)\\
& \geq f^{-1}\left[\sum_i p_i f(l_i)\right] + \epsilon \cdot \left[\sum_i p_i f'(l_i)\right] \cdot (f^{-1})' \left[\sum_i p_i f(l_i)\right] & (4.8)\\
& = f^{-1}\left[\sum_i p_i f(l_i) + \epsilon \cdot \sum_i p_i f'(l_i)\right] + O(\epsilon^2) & (4.9)\\
& = f^{-1}\left[\sum_i p_i f(l_i + \epsilon) + O(\epsilon^2)\right] + O(\epsilon^2) & (4.10)\\
& = f^{-1}\left[\sum_i p_i f(l_i + \epsilon)\right] + O(\epsilon^2). & (4.11)
\end{aligned}$$

Therefore,

$$\begin{aligned}
f^{-1}\left[\sum_i p_i f(l_i + c)\right] & \leq \epsilon + f^{-1}\left[\sum_i p_i f(l_i + c - \epsilon)\right] + O(\epsilon^2) & (4.12)\\
& \leq \cdots & (4.13)\\
& \leq \epsilon \left\lfloor \frac{c}{\epsilon} \right\rfloor + f^{-1}\left[\sum_i p_i f(l_i) + c - \epsilon \left\lfloor \frac{c}{\epsilon} \right\rfloor\right] + O(\epsilon) & (4.14)\\
& \leq c + f^{-1}\left[\sum_i p_i f(l_i)\right] + O(\epsilon) & (4.15)
\end{aligned}$$

so that, taking $\epsilon \to 0$, $f^{-1}[\sum_i p_i f(l_i + c)] \leq c + f^{-1}[\sum_i p_i f(l_i)]$. Thus, knowledge that $f'[f^{-1}(\sum_i p_i f'(l_i))] \geq \sum_i p_i f'(l_i)$ is sufficient to know F is subtranslatory.

Given the same regularity conditions we initially assumed, equations (4.7–4.11) reverse, resulting in the converse. ■

Thus, for such f, we have the bounds of equation (4.5) for the optimum value of the penalty function. We already saw examples relevant for this theorem. Another example, previously mentioned, is the Campbell case of $f(x) = \alpha x + \beta x^2$, for $\alpha, \beta \geq 0$, which we may show to also be subtranslatory. In this case, we have

$$\begin{aligned} f'(x) &= \alpha + 2\beta x & (4.16) \\ f^{-1}(x) &= \sqrt{\left(\frac{\alpha}{2\beta}\right)^2 - \frac{x}{\beta}} - \frac{\alpha}{2\beta} & (4.17) \\ F(L_{\mathcal{X}}, \boldsymbol{p}) &= \sqrt{\left(\frac{\alpha}{2\beta}\right)^2 + \sum_i p_i \left(\frac{\alpha}{\beta} l_i + l_i^2\right)} - \frac{\alpha}{2\beta}. & (4.18) \end{aligned}$$

We may achieve the desired inequality as follows:

$$\begin{aligned} \sum_i p_i l_i^2 &\geq \left(\sum_i p_i l_i\right)^2 & (4.19) \\ \alpha^2 + 4\alpha\beta \sum_i p_i l_i + 4\beta^2 \sum_i p_i l_i^2 &\geq \alpha^2 + 4\alpha\beta \sum_i p_i l_i + 4\beta^2 \left(\sum_i p_i l_i\right)^2 & (4.20) \\ \sqrt{\alpha^2 + 4\beta \sum_i p_i(\alpha l_i + \beta l_i^2)} &\geq \sum_i p_i(\alpha + 2\beta l_i) & (4.21) \\ f'[F(L_{\mathcal{X}}, \boldsymbol{p})] &\geq \sum_i p_i f'(l_i) & (4.22) \end{aligned}$$

We thus have an important property for cases of interest, even those which are not convex. We will find that for the generalized quasilinear convex coding problem, an efficient coding algorithm may be used to find the optimal integer solution. Before we solve this or any special cases, however, let us discuss a related problem, the *Coin Collector's problem.*

4.3 The Coin Collector's problem

4.3.1 Problem statement and properties

Recall that $2^{\mathbb{Z}}$ denotes the set of all integer powers of two. The Coin Collector's problem of size m considers m "coins" with width $\rho_i \in 2^{\mathbb{Z}}$; one can think of width as coin face value, e.g., $\rho_i = \frac{1}{4}$ for a quarter. Each coin also has weight $\mu_i \in \mathbb{R}$. The final problem parameter is total width, denoted t. The problem is then:

$$\begin{array}{ll} \text{Minimize}_{\{B \subseteq \mathcal{S}(m)\}} & \sum_{i \in B} \mu_i \\ \text{subject to} & \sum_{i \in B} \rho_i = t, \end{array} \tag{4.23}$$

where $\mathcal{S}(m) \triangleq \{1, \ldots, m\}$.

We thus wish to fit exactly total width t coins in a minimum weight "container." This problem is an input-restricted case of the knapsack problem, which, in general, is $\mathcal{NP}$-hard [32, 66]. For the knapsack problem, then, no polynomial time algorithms are known, only pseudo-polynomial ones [32]. However, we present a linear time solution to (4.23), first proposed in [59], called the *Package-Merge Algorithm.*

We illustrate and prove a slightly simplified version of the version of the algorithm introduced in [59]. In our notation, we use $i \in \mathcal{S}(m)$ to denote both the index of a coin and the coin itself, and $\mathcal{I}$ to represent the m items along with their weights and widths. The optimal solution of the problem is a function of total width t and items $\mathcal{I}$. We therefore denote this solution as $\mathrm{CC}(\mathcal{I}, t)$ (read, "the [optimal] coin collection for $\mathcal{I}$ and t").

Note that we assume the solution exists but may not be unique. In the case of nonunique solutions, tie resolution for $\arg\min$'s may for now be random or deterministic; we expand further on this later. (A modified version of the algorithm considers the case where a solution may not exist, and another modification allows for the equality in (4.23) to be replaced by an inequality, but neither is needed here.) Due to solution existence, $t = t_n/t_d$ for some unique odd $t_n \in \mathbb{Z}$ and $t_d \in 2^{\mathbb{Z}}$. (For the purposes of this exposition, the "numerator" and "denominator" refer to the unique pair of an odd integer and a power of two, respectively, which, divided, form t. This power of two may be noninteger.)

4.3.2 Package-Merge algorithm and proof of correctness

Algorithm variables

At any point in the algorithm, assume the following definitions:

Remainder	t_s	$\triangleq$	the unique $t_d \in 2^{\mathbb{Z}}$ such that $\frac{t}{t_d}$ is an odd integer
Minimum width	ρ^*	$\triangleq$	$\min_{i\in\mathcal{I}} \rho_i$ (note $\rho^* \in 2^{\mathbb{Z}}$)
Small width set	$\mathcal{I}^*$	$\triangleq$	$\{i \mid \rho_i = \rho^*\}$ (by definition, $\lvert\mathcal{I}^*\rvert \geq 1$)
"First" item	i^*	$\triangleq$	$\arg\min_{i\in\mathcal{I}^*} \mu_i$
"second" item	i^{**}	$\triangleq$	$\begin{cases} \arg\min_{i\in\mathcal{I}^*\backslash\{i^*\}} \mu_i, & \lvert\mathcal{I}^*\rvert > 1 \\ \Lambda \text{ (a null value)}, & \lvert\mathcal{I}^*\rvert = 1. \end{cases}$

Then the following is a recursive description of the algorithm:

Recursive Package-Merge Procedure [59]

Basis. $t = 0$. $\mathrm{CC}(\mathcal{I}, t)$ is the empty set.

Case 1. $\rho^* = t_s$ *and* $\mathcal{I} \neq \emptyset$: $\mathrm{CC}(\mathcal{I}, t) = \mathrm{CC}(\mathcal{I}\backslash\{i^*\}, t - \rho^*) \cup \{i^*\}$.

Case 2a. $\rho^* < t_s$, $\mathcal{I} \neq \emptyset$ *and* $\lvert\mathcal{I}^*\rvert = 1$: $\mathrm{CC}(\mathcal{I}, t) = \mathrm{CC}(\mathcal{I}\backslash\{i^*\}, t)$.

Case 2b. $\rho^* < t_s$, $\mathcal{I} \neq \emptyset$ *and* $\lvert\mathcal{I}^*\rvert > 1$: Let i' be a new item with weight $\mu_{i'} = \mu_{i^*} + \mu_{i^{**}}$ and width $\rho_{i'} = \rho_{i^*} + \rho_{i^{**}} = 2\rho^*$. This new item is thus a combined item, or *package*, formed by combining items i^* and i^{**}. Let $S' = \mathrm{CC}(\mathcal{I}\backslash\{i^*, i^{**}\} \cup \{i'\})$ (the optimization of the packaged version). If $i' \in S'$, then $\mathrm{CC}(\mathcal{I}, t) = S'\backslash\{i'\} \cup \{i^*, i^{**}\}$; otherwise, $\mathrm{CC}(\mathcal{I}, t) = S'$.

Proof: We show that the Package-Merge algorithm produces an optimal solution via induction on the depth of the recursion. The basis is trivially correct, and each inductive case reduces the number of items by one. The inductive hypothesis on $t_s \geq 0$ and $\mathcal{I} \neq \emptyset$ is that the algorithm is correct for any problem instance that requires fewer recursive calls than instance $(\mathcal{I}, t)$.

If $\rho^* > t_s > 0$, or if $\mathcal{I} = \emptyset$ and $t \neq 0$, then there is no solution to the problem, contrary to our assumption. Thus all feasible cases are covered by those given in the algorithm. Case 1 indicates that the solution must contain an odd number of elements of width ρ^*. These must include the minimum weight item in $\mathcal{I}^*$, since otherwise we could substitute one of the items with this "first" item and achieve improvement. Cases 2 indicates that the solution must contain an even number of elements of width ρ^*. If this number is 0, neither i^* nor i^{**} is in the solution. If it is not, then they both are. If $i^{**} = \Lambda$, the number is 0, and we have Case 2a. If not, we may "package" the items, considering the replaced package as one item, as in Case 2b. Thus the inductive hypothesis holds and the algorithm is correct. ■

Figure 4.1 presents a simple example of this algorithm at work, finding minimum total weight items of total width $t = 3$ (or, in binary, 11_b). In the figure, item width represents numeric width and item area represents numeric weight. First, the minimum weight item with width $\mu_{i^*} = t_s = 1$ is put into the solution set. Then, the remaining minimum width items are packaged into a merged item of width 2 (10_b). Finally, the minimum weight item/package with width $\mu_{i^*} = t_s = 2$ is added to complete the solution set, which is now of weight 6. The remaining packaged item is left out in this case; when the algorithm is used for coding, several items may be left out of the optimal set.

An iterative linear time implementation appears in [59].

4.4 Overall algorithm

We now find a reduction from the generalized quasilinear convex coding problem to the Coin Collector's problem. We first assume bounds on the maximum code length of possible solutions. This may be the maximum unary codeword length of $n-1$. Alternatively, it may be explicit in the definition of the problem. Consider the length-limited coding problem of [59],

$$f(l_i, p_i) = \begin{cases} p_i l_i, & l_i \leq l_{max} \\ +\infty, & l_i > l_{max} \end{cases} \tag{4.24}$$

for some fixed $l_{max} \geq \lceil \lg n \rceil$. This is upper bounded by l_{max}. A third possibility is that maximum length may be implicit in some property of the set of optimal solutions [13, 50]; we explore this in Section 4.6.

Therefore we may restrict ourselves to codes with n codewords, none of which has

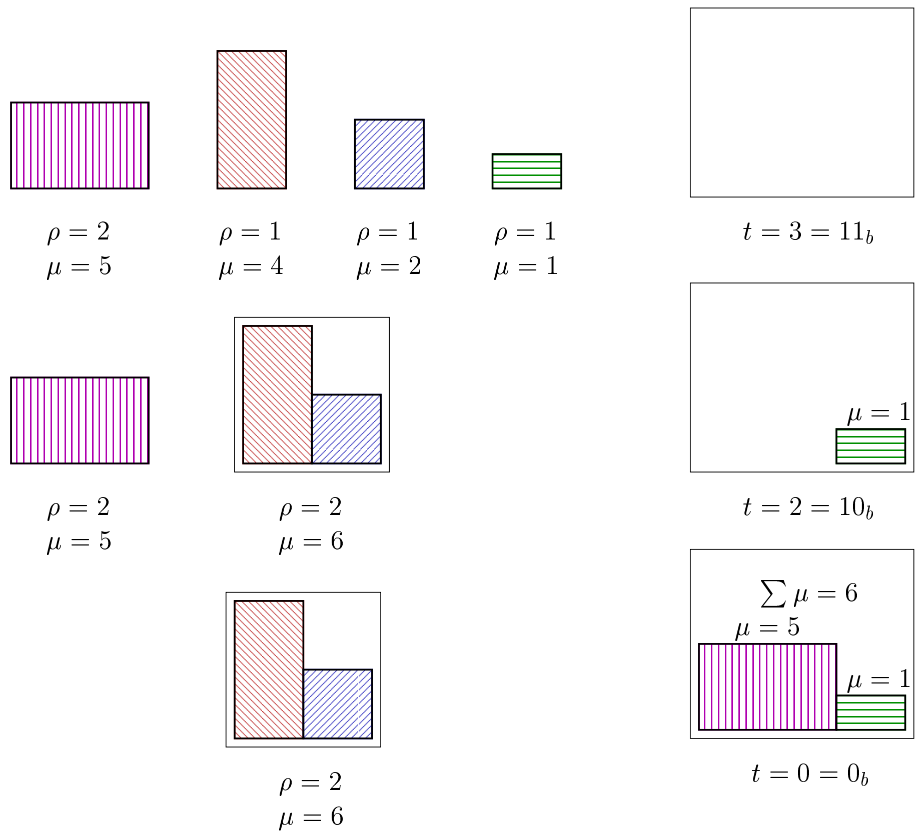


Figure 4.1: A simple example of the Package-Merge algorithm

greater length than l_{max} for some $\lceil \lg n \rceil \leq l_{max} \leq n - 1$. With this we may now introduce *nodeset* notation as an alternative method of code representation to binary code trees:

Definition 21 *A* node *is an ordered pair of integers (i, l) such that $i \in \{1, \ldots, n\}$ and $l \in \{1, \ldots, l_{max}\}$. Call the set of all nl_{max} possible nodes — usually arranged in a grid — I (see Figures 4.2 and 4.3). The set of nodes, or* nodeset, *corresponding to a codeword c_i (length l_i) is the set of the first l_i nodes of column i, that is,* $\text{nodeset}(c_i) = \text{nodeset}(l_i) \triangleq \{(j, l) \mid j = i,\ l \in \{1, \ldots, l_i\}\} \subseteq I$*. The nodeset corresponding to length ensemble $L_{\mathcal{X}}$ is* $\text{nodeset}(L_{\mathcal{X}}) \triangleq \bigcup_i \text{nodeset}(l_i)$*, also $\subseteq I$; this corresponds to a set of n codewords, a code. If node $(i, l) \in I$ then we say it has* width $\rho(i, l) \triangleq 2^{-l}$ *and* weight $\mu(i, l) \triangleq f(l, p_i) - f(l - 1, p_i)$*, as in Figure 4.2.*

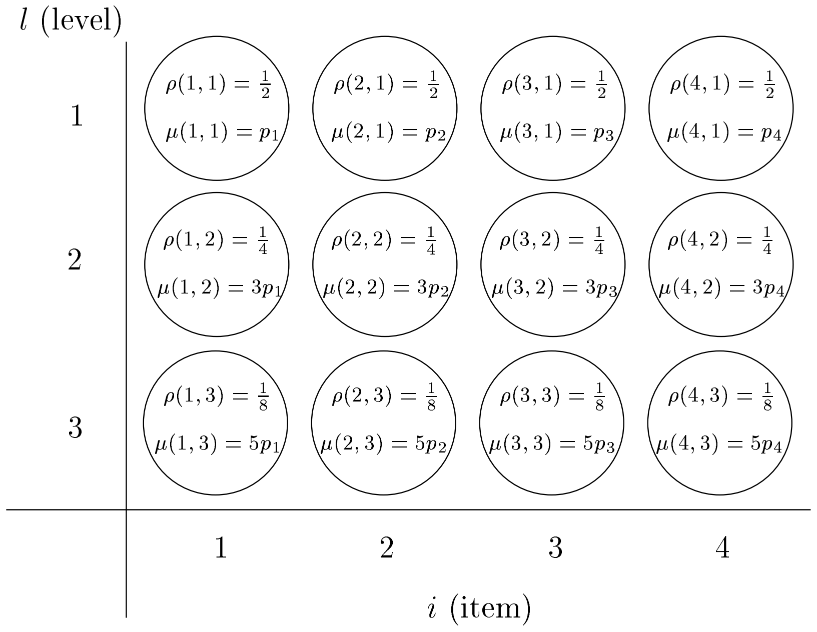


Figure 4.2: The set of nodes I with widths ($\rho(i,l)$'s) and weights ($\mu(i,l)$'s) for $f(l,p) = pl^2$, $n = 4$, $l_{max} = 3$

If a subset N of all nodes I is a valid nodeset, then it is straightforward to find the corresponding length ensemble and thus a code. Any optimal solution N of the Coin Collector's problem for $t = n-1$ on coins $\mathcal{I} = I$ is a nodeset for an optimal solution of the coding problem. Thus we have a suitable method of solving the generalized quasilinear convex penalty.

To show this reduction, define first for any $N = \text{nodeset}(L_{\mathcal{X}})$:

$$\begin{aligned} \rho(N) &\triangleq \sum_{(i,l)\in N} \rho(i,l) && (4.25)\\ &= \sum_{l_i \in L_{\mathcal{X}}} 1 - 2^{-l_i} && (4.26)\\ &= n - \sum_{l_i \in L_{\mathcal{X}}} 2^{-l_i} && (4.27) \end{aligned}$$

which is in $[n-1, n)$ for prefix-free codes due to the Kraft inequality. Kraft is satisfied with equality on the left end of this interval. Also define:

$$\begin{aligned}
\mu(N) &\triangleq \sum_{(i,l)\in N} \mu(i,l) && (4.28)\\
&= \sum_{i=1}^{n} \left\{ f(0,p_i) + \sum_{l=1}^{l_i} [f(l,p_i) - f(l-1,p_i)] \right\} && (4.29)\\
&= \sum_{i=1}^{n} f(l_i,p_i) && (4.30)\\
&= F(L_{\mathcal{X}}, \boldsymbol{p}) && (4.31)
\end{aligned}$$

since $f(0,p_i) = 0 \ \forall \ i \in \mathcal{X}$. This establishes the parallel between the Coin Collector's problem and this coding problem. To prove the reduction, we need the following lemma:

Lemma 5 *Suppose that N is a nodeset of width $x2^{-l} + r$ where x is an integer and $0 < r < 2^{-l}$. Then N has a subset R with width r.*

Proof: As with the proof of correctness for the Package-Merge algorithm, use induction on the cardinality of the set. The base case $|N| = 1$ is trivial since then $x = 0$. Assume the lemma holds for all $|N| < n$, and suppose $|\tilde{N}| = n$. Let $i^* = \arg\min_{i\in\tilde{N}} \mu_i$ and $k = \min_{i\in\tilde{N}} \mu_i$. We have $k > l$ since $0 < r < 2^{-l}$. Then r must be an integer multiple of 2^{-k}. If $r = 2^{-k}$, $R = \{i^*\}$ is a solution. Otherwise let $N' = \tilde{N}\backslash\{i^*\}$ (so $|N'| = n-1$) and let R' be the subset obtained from solving the lemma for set N' of width $r - 2^{-k}$. Then $R = R' \cup \{i^*\}$. ■

We may now prove the main theorem:

Theorem 9 *Any $N \subseteq I$ that is a solution of the Coin Collector's problem for $t = \rho(N) = n-1$ has a corresponding $L_{\mathcal{X}}^N$ such that $N =$ nodeset$(L_{\mathcal{X}}^N)$ and $\rho(N) = \min_{L_{\mathcal{X}}} F(L_{\mathcal{X}}, \boldsymbol{p})$.*

Proof: By monotonicity, any optimal solution satisfies the Kraft inequality with equality. Thus all optimal length ensemble nodesets have $\rho(\text{nodeset}(L_{\mathcal{X}})) = n-1$.

Suppose N is a solution to the Coin Collector's problem but is not a valid nodeset of a length ensemble. Then there exists an $(i,l) \in I$ with $l > 1$ such that $(i,l) \in N$ and $(i,l-1) \in I \backslash N$. Let $R' = N \cup \{(i,l-1)\} \backslash \{(i,l)\}$. Then $\rho(R') = n - 1 + 2^{-l}$ and, due to convexity, $\mu(R') \leq F(L_{\mathcal{X}}^N, \boldsymbol{p})$. Thus, using Lemma 5, there exists an $R \subset R'$ such that $\rho(R) = n-1$ and $\mu(R) < \mu(R') \leq F(L_{\mathcal{X}}^N, \boldsymbol{p})$. However, since we assumed N to be an optimal solution of the Coin Collector's problem, this is a contradiction, and thus any solution corresponds to an (optimal) length ensemble. ■

Thus we have an algorithm that finds an optimal code in $O(nl_{max})$ time for any $f(l_i, p_i)$ that is monotonically increasing and convex. (Note that the generality of this algorithm makes this extendable to problems of the form $\sum_i f_i(l_i, p_i)$ for n different functions f_i. This might be applicable if we desire a nonlinear weighting for codewords in addition to and possibly independent of codeword length and probability.)

The complexity of the algorithm in terms of n alone depends on the structure of f and $\boldsymbol{p}$. To show how, we first need some definitions:

Definition 22 *The space of problems being considered is called a* flat class *if, for any solution $L_{\mathcal{X}}$, $\frac{\max_i l_i}{\lg n} < \kappa$ for some known constant κ. For example, the space of linear Huffman coding problems for $p_n \geq \frac{1}{2n}$ is a flat class. (This may be shown using [13].)*

Definition 23 *A problem is* differentially monotonic *in $\boldsymbol{p}$ if $\forall\ l > 1,\ p_i > p_j \Rightarrow [f(l,p_i) - f(l-1,p_i)] > [f(l,p_j) - f(l-1,p_j)]$ unless $f(l-1,p_i) = +\infty$.*

The latter property implies f is continuous in $\boldsymbol{p}$ at all but a countable number of points, but here we only consider cases in which it is continuous everywhere. With the exception of the f for minimal DABR when $b + d \leq -1$, all f considered in this paper are differentially monotonic and continuous in $\boldsymbol{p}$, including all penalties of expectation (those of the form $E_{\boldsymbol{p}}[f(l(X))]$).

If the problem space considered is a flat class, l_{max} is $O(\log n)$; it is $O(n)$ in general. Sorting items for the Package-Merge algorithm is $O(n \log n)$ if we are guaranteed that $l_i \leq l_k\ \forall\ i < k$, and $O(nl_{max} \log n)$ in other cases. Thus problem complexity for this solution ranges from $O(n \log^2 n)$ (flat) to $O(n^2 \log n)$ (others) with space requirement $O(n \log^2 n)$ (flat) to $O(n^2)$ (others). In differentially monotonic cases, however, this can be improved upon.

4.5 A linear space algorithm

Note that the resulting length ensemble need not have the property that if $i < k$, then $l_i \leq l_k$ (the first property of Lemma 1). For example, if $p_i = p_k$, we are guaranteed no particular inequality relation between l_i and l_k. Also, even if all p_i were distinct, there are cases for which we would expect the inequality relation reversed from the linear case. An example of this is $f(l_i, p_i) = p_i^{-1} 2^{l_i}$, which represents no practical problem, although, if we exclude the domain constraints, it corresponds to (-1.5)-average 1-redundancy, $b = -1.5$ and $d = 1$ in equation (3.3).

The problem space for the above algorithm is quite general, but we can improve upon the algorithm by insisting that the problem be differentially monotonic and all entries p_i in $\boldsymbol{p}$ be distinct, a condition later relaxed. The resulting algorithm uses only linear space. First we need a definition,

Definition 24 *A* monotonic *nodeset, N, is one with the following properties:*

$$(i, l) \in N \Rightarrow (i+1, l) \in N, \quad \text{for } i < n, \tag{4.32}$$

$$(i, l) \in N \Rightarrow (i, l-1) \in N, \quad \text{for } l > 1. \tag{4.33}$$

An example of an optimal monotonic nodeset is the set of nodes enclosed by the dashed line in Figure 4.3. Note that a nodeset is monotonic if and only if it corresponds to a length ensemble with lengths sorted increasing order. Note too that this order implies a unique tree with leaf nodes in this order, and thus a unique code. We may thus refer to such trees and codes as monotonic as well. For example, tree (c) of Figure 2.5 is monotonic while tree (a) and tree (b) are not.

Lemma 6 *If a problem is differentially monotonic in $\boldsymbol{p}$ and monotonically increasing and convex in l_i's, and if $\boldsymbol{p}$ has no repeated values, then $N = \mathrm{CC}(I, n-1)$ is monotonic.*

Proof: The second property of monotonic nodesets (4.33) was proved in Theorem 9. Suppose that we can find an N that violates the first property. Then there exists $j \neq k$ such that $p_j < p_k$ and $l_j < l_k$ for optimal codeword lengths $L_{\mathcal{X}}$ ($N = \text{nodeset}(L_{\mathcal{X}})$). Consider $L'_{\mathcal{X}}$ with lengths for items j and k interchanged. Then

$$\begin{aligned}
F(L'_{\mathcal{X}}) - F(L_{\mathcal{X}}) &= \sum_i f(l'_i, p_i) - \sum_i f(l_i, p_i) && (4.34)\\
&= [f(l_k, p_j) - f(l_j, p_j)] - [f(l_k, p_k) - f(l_j, p_k)] && (4.35)\\
&= \sum_{l=l_j+1}^{l_k} \{[f(l, p_j) - f(l-1, p_j)] - [f(l, p_k) - f(l-1, p_k)]\} && (4.36)\\
&< 0 && (4.37)
\end{aligned}$$

where equation (4.37) is due to differential monotonicity. However, this implies that $F(L_{\mathcal{X}})$ is not an optimal code, and thus we cannot have an optimal nodeset without monotonicity unless values in $\boldsymbol{p}$ are repeated. ■

Taking advantage of this relation to trade off a constant factor of time for drastically reduced space complexity has been done in [58] for the case of the length-limited (linear) penalty of equation (4.24). We now extend this to cases monotonically increasing and convex in l_i's and differentially monotonic in $\boldsymbol{p}$.

At a given point in the Package-Merge algorithm for the coding problem, because the algorithm operates one level at a time, fewer than $2n$ packages are kept in memory. Each package may have several nodes, however, hence the space complexity of nl_{max}. The idea of reducing space complexity is to keep only four attributes of each package in memory instead of the full contents. In this manner, we retain enough information to reconstruct the optimal nodeset in algorithmic postprocessing.

Define $l_{mid} \triangleq \lfloor \frac{l_{max}+1}{2} \rfloor$. Package attributes allow us to divide the problem into two subproblems with total complexity not exceeding that of half the original problem. Consider a package S. For S, we retain the attributes that follow:

1. Weight: $\mu(S) \triangleq \sum_{s \in S} \mu_s$

2. Width: $\rho(S) \triangleq \sum_{s \in S} \rho_s$

3. Midct: $\nu(S) \triangleq |S \cap L_{mid}|$

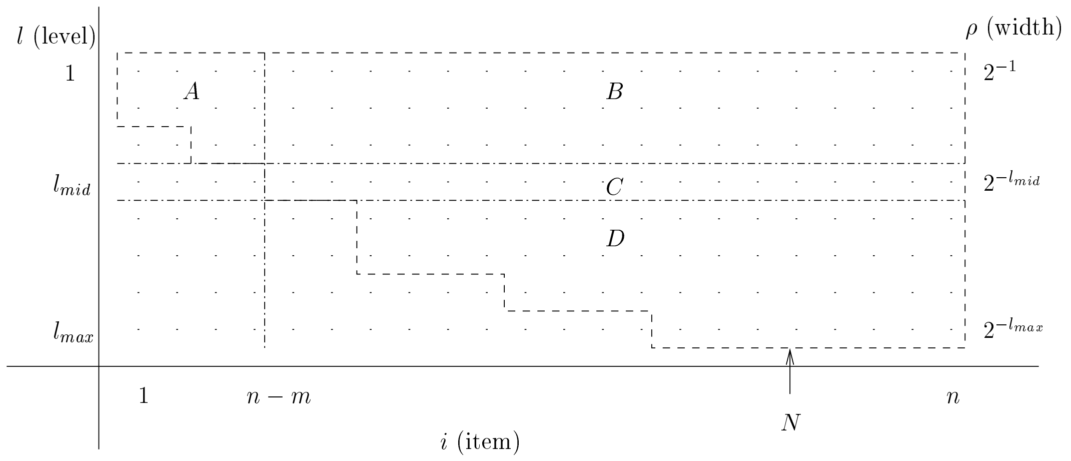


Figure 4.3: The set of nodes I, an optimal nodeset N, and disjoint subsets A, B, C, D

4. Hiwidth: $\psi(S) \triangleq \sum_{s\in S\cap L_{hi}} \rho_s$

where $L_{hi} \triangleq \{(i,l) \mid l > l_{mid}\}$, $L_{mid} \triangleq \{(i,l) \mid l = l_{mid}\}$, and $L_{lo} \triangleq \{(i,l) \mid l < l_{mid}\}$.

This retains enough information to complete the "first run" of the algorithm with $O(n)$ space. The result will be the package attributes for the optimal nodeset N. Thus, at the end of this first run, we know the value for $m = \nu(N)$, and we can consider N as the disjoint union of four sets, shown in Figure 4.3:

1. A = nodes in $N \cap L_{lo}$ with indices in $[1, n-m]$,

2. B = nodes in $N \cap L_{lo}$ with indices in $[n-m+1, n]$,

3. C = nodes in $N \cap L_{mid}$,

4. D = nodes in $N \cap L_{hi}$.

Due to monotonicity of N, it is trivial that $C = [n-m+1, n] \times \{l_{mid}\}$ and $B = [n-m+1, n]\times[1, l_{mid}-1]$. Note then that $\rho(C) = m2^{-l_{mid}}$ and $\rho(B) = m[1-2^{-(l_{mid}-1)}]$. Thus we need merely to recompute which nodes are in A and D.

Since $D \subseteq L_{hi}$, $\rho(D) = \psi(N)$ and $\rho(A) = \rho(N) - \rho(B) - \rho(C) - \rho(D)$. Given their respective widths, A is a minimal weight subset of $[1, n-m] \times [1, l_{mid}-1]$ and

D is a minimum weight subset of $[n-m+1,n]\times[l_{mid}+1,l_{max}]$. The nodes at each level of A and D may be found by recursive calls to the algorithm. In doing so, we use only $O(n)$ space. We now prove that the time complexity remains the same:

Theorem 10 *The above recursive version of generalized quasilinear convex coding has $O(nl_{max})$ time complexity.*

Proof: To find the time complexity, set up the following recurrence relation: Let $\tau(n,l)$ be the worst case time to find the minimal weight subset N of $[1,n]\times[1,l]$ of a given width t, assuming N is monotonic. Then there exist constants c_1 and c_2 such that

$$\tau(n,l) \le c_1 n, \quad \text{for} \quad l<3, \tag{4.38}$$

$$\tau(n,l) \le c_2 nl + \tau(n^{(1)},l^{(1)}) + \tau(n^{(2)},l^{(2)}), \quad \text{for} \quad l \ge 3, \tag{4.39}$$

where $l^{(1)} = l_{mid} - 1 \le \lfloor \frac{l}{2} \rfloor$, $l^{(2)} = l - l^{(1)} - 1 \le \lfloor \frac{l}{2} \rfloor$, and an adversary chooses $n^{(1)} + n^{(2)} = n$. Note that $\tau(n,l) = O(\upsilon(n,l))$, where υ is any function satisfying the recurrence

$$\upsilon(n,l) \ge c_1 n, \quad \text{for} \quad l<3, \tag{4.40}$$

$$\upsilon(n,l) \ge c_2 nl + \upsilon\left(n^{(1)},\frac{l}{2}\right) + \upsilon\left(n-n^{(1)},\frac{l}{2}\right), \quad \text{for} \quad l \ge 3, \tag{4.41}$$

which $\upsilon(n,l) = (c_1+2c_2)nl$ does. Thus, time complexity is $O(nl_{max})$. ■

The overall complexity is linear space and $O(nl_{max})$ time — $O(n \log n)$ considering only flat classes, $O(n^2)$ in general.

However, the assumption of distinct p_i puts an undesirable restriction on our input. In their original algorithm from [59], Larmore and Hirschberg suggest modifying the probabilities slightly to make them distinct, but this is inelegant and the resulting algorithm has the additional drawback of being nondeterministic. A variant of this approach involves using changes by a suitably small variable $\epsilon > 0$ to make identical values distinct. However, here we present a more elegant alternative, still deterministic, and applicable to all differentially monotonic cases.

In initial ordering, we sort the items m_i in reverse of their order of appearance, i.e., descending as $m_n, m_{n-1}, \ldots, m_1$. Recall $\boldsymbol{p}$ is a nonincreasing vector. In the first stage of the Package-Merge algorithm, then, combined items are paired off such that all similar-width items in one package have adjacent indices. Recall that packages of items will be either in the final nodeset or absent from it as a whole. In addition, we choose nonmerged items over merged in the case of ties, in the same manner as in the two-queue bottom-merge method of Huffman coding [91]. We obtain a deterministic algorithm retaining this adjacency, and along with it width order preference for items of equal weight at all steps. Thus we no longer need to worry about a case where $i < k$ but $l_i > l_k$. An additional benefit is that, in the case of ties, this results in minimizing maximal length among optimal codes, as with bottom-merge Huffman coding in [84].

4.6 Further algorithmic optimization

Above we assumed we knew a maximum bound for length, although in the overall complexity analysis we assumed this was $n-1$. We now explore a method for finding better upper bound and thus a more efficient algorithm. First two definitions due to Larmore in [58]:

Definition 25 *The* weight function *corresponding to penalty $f(l,p)$ is $M_f(l,p) = f(l,p) - f(l-1,p)$.*

Definition 26 *Consider penalty functions f and g. We say that g is* flatter *than f if $l' > l \Rightarrow M_g(l,p)M_f(l',p') \leq M_f(l,p)M_g(l',p')$.*

A consequence of the Convex Hull Theorem of [58] is that, given g flatter than f, for any $\boldsymbol{p}$, there exist f-optimal $L_{\mathcal{X}}^{(f)}$ and g-optimal $L_{\mathcal{X}}^{(g)}$ such that $L_{\mathcal{X}}^{(f)}$ is lexicographically greater than $L_{\mathcal{X}}^{(g)}$. This explains why the word "flatter" is used.

Penalties flatter than the linear penalty may therefore yield a useful upper bound, simplifying complexity. All convex Campbell penalties are flatter than the linear penalty. (There are some generalized quasilinear convex coding penalties that are not flatter than the linear penalty (e.g., $f(l,p) = lp^2$) and some flatter penalties that are not Campbell/quasilinear (e.g., $f(l,p) = 2^l(p + 0.1 \sin \pi p)$), so no other similarly straightforward relation exists.)

Thus for most penalties we have considered, due to the aforementioned consequence, we may use the results of a pre-algorithmic Huffman coding of the items, or the upper bounds of Buro in [13], to find an upper bound on codeword length. Using the maximum unary codeword length of $n-1$ and techniques involving the Golden Mean, $\Phi \triangleq \frac{\sqrt{5}+1}{2}$, Buro gives the upper limit of length for a (standard) Huffman codeword as

$$\min\left\{ \left\lfloor \log_\Phi \left(\frac{\Phi + 1}{p_n \Phi + p_{n-1}} \right) \right\rfloor, n-1 \right\}, \tag{4.42}$$

which would thus be an upper limit on codeword length for the minimal optimal code obtained using any flatter penalty function, such as a convex Campbell function. This may be used to decrease complexity, especially in a case in which we encounter a flat class of problem inputs.

In addition to this, we may be able to adapt the techniques for length-limited Huffman coding of Moffat et al. in [48, 63, 74, 88, 89], to improve upon our algorithm. We do not explore these, however, as these cannot improve asymptotic results with the exception of a few special cases. Unfortunately, other approaches to length-limited Huffman coding with improved algorithmic complexity [9, 83] cannot extend to nonlinear penalties.

4.7 Examples

The generalized quasilinear convex coding problem includes the minimal DABR generalization (3.3) of the exponential case for $d \in [0, +\infty)$. It also includes such cases as the length-limited coding of (4.24), the ath moment about zero

$$f(l_i, p_i) = p_i l_i^a \tag{4.43}$$

for $a \geq 1$, and the aforementioned case of

$$f(l_i, p_i) = p_i(\alpha l_i + \beta l_i^2) \tag{4.44}$$

for $\alpha, \beta > 0$. These last three penalties are convex Campbell problems (2.2) and thus differentially monotonic and flatter than the linear penalty.

4.8 Extending to more exotic penalties

We previously mentioned that [58] used a quadratic penalty in order to find a code minimizing a convex but nonquasilinear function corresponding to a delay penalty, the expected waiting time between information request and receipt. For the statement of the problem and properties, let $\bar{l} \triangleq E_{\boldsymbol{p}}[l(X)]$ and $\bar{l^2} \triangleq E_{\boldsymbol{p}}[l(X)^2]$ for a given $L_{\mathcal{X}}$, and let $\bar{l^*} \triangleq \min_{L_{\mathcal{X}}} \bar{l}$, the minimized value in standard (Huffman) coding. The delay penalty is then given by

$$F(L_{\mathcal{X}}, \boldsymbol{p}) = \begin{cases} \frac{\lambda \bar{l^2}}{2(1-\lambda \bar{l})} + \bar{l}, & \bar{l} < \frac{1}{\lambda} \\ +\infty, & \bar{l} \geq \frac{1}{\lambda} \end{cases} \tag{4.45}$$

where λ is the expected number of requests in a channel per unit time, unit time being chosen to correspond with the time to transmit one bit [26]. This formula is a result of a model that assumes the message must pass through an $M/G/1$ queue, i.e., that the requests are memoryless (Poisson) and we have only one device to service them; thus the Pollaczek-Khinchin formula applies [30,51,79]. Service time is assumed to be proportional to codeword length. (In the analogous problem, the linear penalty minimizes delay for an $M/G/\infty$ queue.) Thus the problem is well-posed iff $\lambda \in (0, \bar{l^*}^{-1})$. Our improvement of (4.44) improves the algorithmic performance in finding the optimal value from $O(n^5)$ to $O(n^4)$.

Let us note some structure to the solution of this problem. First, note that for both light traffic ($\lambda \downarrow 0$) and heavy traffic ($\lambda \uparrow \bar{l^*}^{-1}$), the optimal solution is minimum variance Huffman coding, i.e., bottom-merge Huffman coding. For light traffic, $F(L_{\mathcal{X}}, \boldsymbol{p}) \approx \bar{l} + \frac{\lambda}{2}\bar{l^2}$, so minimum variance coding should be used, as in equation (2.7). For heavy traffic, as noted by Flores in [26], $F(L_{\mathcal{X}}, \boldsymbol{p}) < +\infty$ only for optimal Huffman codes and F is minimized with minimum variance.

We might be tempted to conclude that minimum variance Huffman coding is sufficient to solve the general case, but this is not true. Consider the two possible monotonic code trees with four leaf nodes, the unary tree $L_u = \{1\ 2\ 3\ 3\}$ and the flat tree $L_f = \{2\ 2\ 2\ 2\}$. Let $\boldsymbol{p} = (0.35, 0.35, 0.15, 0.15)$. The unary tree is optimal for Huffman coding and thus for high and low traffic cases, but the flat tree has minimal delay iff $\lambda \in [\frac{169-\sqrt{4721}}{596}, \frac{169+\sqrt{4721}}{596}] \approx [0.1683, 0.3988]$ (requests per symbol). For example, if $\lambda = 0.250$, $F(L_u) = \frac{2489}{820} \approx 3.035$ and $F(L_f) = 3$, so the flat code is optimal.

While we may wish to generalize to arbitrary convex functions, there is no general solution known which is more efficient than general techniques for such strongly $\mathcal{NP}$-hard problems [32], and the techniques which find precise solutions to these problems have exponential complexity. However, the results of Larmore may be extended to a larger class of problems by combining them with ours. Unfortunately, this class applies to no practical penalty known to us, so we will not prove the case, but instead only comment upon the tools needed to extend Larmore's result.

We want to have a quasiconcave trade-off function between two convex general quasilinear penalties, one of which is flatter than the other.

Definition 27 *Given a real-valued function f defined over the domain $\mathcal{D} \in \mathbb{R}^n$ for some n, that is, $f : \mathcal{D} \to \mathbb{R}$,* superlevel set $S_\alpha(f) \triangleq \{x \in \mathcal{D} \mid f(x) \geq \alpha\}$*. If all superlevel sets are convex, function f is said to be* quasiconcave. Level set $V_\alpha(f) \triangleq \{x \in \mathcal{D} \mid f(x) = \alpha\}$*. Function f is said to be a* trade-off function *if there is no vector $\boldsymbol{w} \in \mathbb{R}^n$ and nonnegative ($\succeq \boldsymbol{0}$) vector $\boldsymbol{v} \neq \boldsymbol{0}$ such that $\boldsymbol{w}, \boldsymbol{w}+\boldsymbol{v} \in V_\alpha(f)$ for some α.*

A subcase of this is the delay problem previously considered. In order to see this, we put the penalty in the form $g(f_1(L_{\mathcal{X}}, \boldsymbol{p}), f_2(L_{\mathcal{X}}, \boldsymbol{p}))$, where $g(x,y) = \frac{\lambda y}{2(1-\lambda x)} + x$, $f_1(L_{\mathcal{X}}, \boldsymbol{p})$ is $\bar{l}$, and $f_2(L_{\mathcal{X}}, \boldsymbol{p})$ is $\overline{l^2}$. The larger set of problems should be soluble in the same manner as the delay problem. If a desirable application is found for this, a formal proof and exact method would be of interest.

4.9 Review of finite alphabet coding cases

Consider again the penalties proposed in Table 1.1. We have found that linear time variants of Huffman coding can solve F_1 through F_7. Penalties F_8 through F_{13} may be solved using the generalized quasilinear convex coding algorithm, and F_{14} may be solved using the result of Larmore, which we reduced from $O(n^5)$ to $O(n^4)$ time and $O(n^3)$ to $O(n^2)$ space; the quadratic space requirement is due to bookkeeping in the overall algorithm [58]. Penalty F_{15} is the problem solved by Humblet in [45] using a reduction to exponential Huffman coding; this minimizes probability of buffer overflow for a $GI/G/1$ buffer of size $\ln a$ where g_T is the moment generating function for interarrival time.

Optimizing the concave penalty of a geometric mean, F_{16}, however, remains open. Also remaining open is how to extend these algorithms to infinite alphabets, a topic we consider in the next chapter.

Chapter 5

Extensions to infinite alphabets

In this chapter, we explore instances of coding for a countably infinite number of codewords. We find a framework for understanding the linear penalty and others. Considering other penalties, we find bounds and an algorithm analogous to those previously found for infinite linear instances.

5.1 Motivation

Now we turn the case where $|\mathcal{X}| = +\infty$. The methods of previous chapters — variants of Huffman coding and the Package-Merge algorithm — are not readily adaptable, as each is bottom-up in nature, starting with the least likely messages and building up. Because of this, we should first investigate the linear penalty before considering any other type.

In [64], Linder, Tarokh, and Zeger show that, for the linear penalty and a fixed $\boldsymbol{p}$, if $H(\boldsymbol{p}) = -\sum_i p_i \lg p_i < +\infty$, a minimum average length code exists. For $H(\boldsymbol{p}) = +\infty$, no code can have finite average length, so no optimum exists for the penalty measure of expected length. However, this measure can be easily modified. We can consider average redundancy (which is equal to Kullback Leibler divergence [21] for binary coding), but it is not trivial to prove that this is always defined. Fortunately, another measure, one explored by Kato, Han, and Nagaoka in [49], yields identical optimal codes as minimal redundancy restricted to cases in which redundancy is defined. Instead of using redundancy, optimal codes are defined as codes for which all compatible finite entropy partitions' codes are optimal. (Recall from Section 2.3 that a compatible partition has a code corresponding to a common-root subtree of the tree for the primary code in question.) For finite entropy, these are equivalent. Because this measure is equivalent to a limit of codes, the results of [64] extend to this penalty.

Because $\mathcal{X}$ is infinite, we cannot define infinite alphabet coding the same way we do for finite alphabets, that is, relating a finite length sequence of inputs to a finite length sequence of outputs. We may still assume the list of inputs is sorted in decreasing order, as the maximum of a countable list with known sum may always be found in finite time. However, we may define two different variations of the problem.

Definition 28 Oracle modeled coding *is the problem of calculating the length of a codeword for item m_i given, via a black box program or oracle, a desired finite subset of $\boldsymbol{p}$ (since this is all we may observe in finite time).* Program modeled coding *is the problem of calculating the length of a codeword for item m_i given a complete specification of $\boldsymbol{p}$; to solve this problem is to be able to calculate any value of l_i using a program (or formula) based on the input program (or formula) completely specifying $\boldsymbol{p}$.*

Recall that, even in the case of infinite alphabets, we can always derive codewords compatible with given codeword lengths, as discussed in Section 2.1. The two aforementioned variations of the problem, however, have a number of differences. In the program variation, we are assumed to have much more information, as we may be able to find useful functions of the entire input — such as the conditional entropy of an arbitrary subset of the outcomes — that would not be available in the infinite case. Because this may be necessary for coding but incalculable in the oracle case, the distinction between program instances and oracle input instances is an important one. Note also that we can obtain an oracle instance from a program one, but not vice versa.

Another difference is that we specify the probability distribution with a finite input in the program case, where in the oracle case we instead converge to the distribution

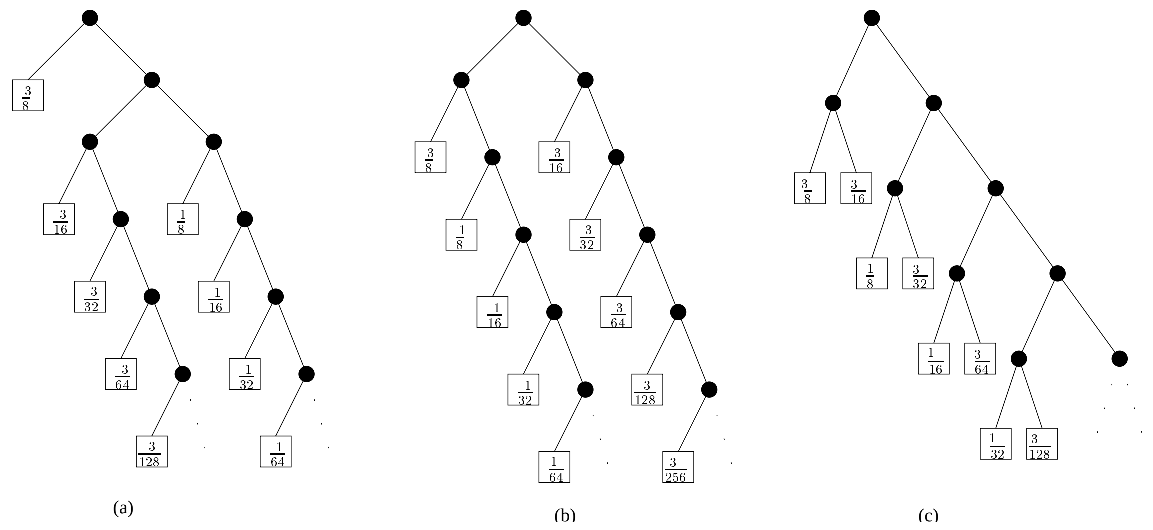


Figure 5.1: Sample coding trees for infinite alphabets

as in [64] via an infinite sequence of programs. Finally, the Kolmogorov complexity — the supremum of the lengths of the programs with $\boldsymbol{p}$ as output — may be finite or infinite in the oracle variation, whereas in the program variation it must be finite. However, because we would expect to encounter optimal coding problems fully formed, not given to us piecewise by some oracle, all practical instances of infinite alphabet coding have finite description.

This is fortunate, as no algorithm can find even the optimum length l_1 of first codeword c_1 for all $\boldsymbol{p}$ given a finite subset of $\boldsymbol{p}$. One can easily see this as follows:

Recall the dyadic geometric probability distribution, $\boldsymbol{p}^{(\boldsymbol{dg})} = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \cdots)$. Consider an event that has one outcome with probability $p_1 = \frac{3}{8}$, any one of an infinite number of distinct dyadic geometrically distributed events with probability $\cup_{k \in \mathbb{N}} \; p_{2k} = \frac{3}{8}$, and any one of an infinite number of distinct dyadic geometrically distributed events with probability $\cup_{k \in \mathbb{N}} \; p_{2k+1} = \frac{1}{4}$. This is $\boldsymbol{p} = (\frac{3}{8}, \frac{3}{16}, \frac{1}{8}, \frac{3}{32}, \frac{1}{16}, \cdots)$, which may have optimal codes with either $l_1 = 1$ or $l_1 = 2$, as in Figure 5.1. Tree (a) and tree (b) are both valid infinite Huffman coding trees for $\boldsymbol{p}$.

Suppose we form a new probability mass function $\boldsymbol{p}^{(\boldsymbol{m},\epsilon)}$ by adding a very small $\epsilon > 0$ to p_m and subtracting ϵ from p_{m+1} for some $m > 1$. If m is odd, $l_1 = 1$ on the optimal code; otherwise $l_1 = 2$. Since m could be arbitrarily large, if the mass function were $\boldsymbol{p}$, there would be no way to use any finite subset of its elements to show it was not $\boldsymbol{p}^{(m,\epsilon)}$ for some (m, ϵ) pair. Thus even the first codeword is incalculable. However, since this problem and its solution may be finitely specified (as we do in our description), the example is codable in the instance of program modeled coding.

Note too that a simple modification with random perturbations of an infinite number of p_i's would have infinite complexity and thus have an incalculable yet existent solution to a problem that cannot be finitely specified; such a problem is thus also incalculable. Returning our attention to Figure 5.1, we see that monotonic tree (c) also results in an optimal code for $\boldsymbol{p}$, one with the same length ensemble as (b), but is not a Huffman coding tree in the sense that (c) has finite subtrees lacking the (strong) sibling property. For trees not lacking this property, this interpretation — which combines the results of Gallager from [29] and Kato, Han, and Nagaoka from [49] — is useful for confirming optimality.

For certain families of oracle modeled instances with finite entropy, there exist algorithms to construct optimal codes. An algorithm suitable for some oracle cases is given by Kato, Han, and Nagaoka in [49]. We explore this result with ideas for generalization to other families. We also consider some program modeled cases. We then generalize to other penalties. First, however, we examine properties of optimal infinite item codes.

5.2 Properties of optimal infinite item codes

Although $L_{\mathcal{X}}$ does not uniquely determine a coding tree (e.g., tree (b) and tree (c) in Figure 5.1), it does determine the number of leaf and internal nodes at each level. This may be iteratively calculated as follows. Label the number of leaf nodes on each level $\lambda_{\mathcal{X}} = \{\lambda_1, \lambda_2, \ldots\}$ and the number of internal nodes $\eta_{\mathcal{X}} = \{\eta_1, \eta_2, \ldots\}$. Let $N_S(i)$ denote the number of i's in sequence S. Then, if we let $\eta_0 = \frac{1}{2}$ for purposes of initialization,

$$\lambda_i = N_{L_{\mathcal{X}}}(i-1), \tag{5.1}$$

$$\eta_i = 2\eta_{i-1} - \lambda_i. \tag{5.2}$$

Thus, if $|\mathcal{X}| < +\infty$, $|\lambda_{\mathcal{X}}| - 1 = |\eta_{\mathcal{X}}| = |L_{\mathcal{X}}|$, and we end calculation when η_i is 0. For $|\mathcal{X}| = +\infty$, $\eta_i > 0 \; \forall \; i \in \mathcal{X}$.

Using this, we may divide infinite alphabet length ensembles into any one of a number of categories. Particularly useful divisions follow:

Definition 29 *We call $L_{\mathcal{X}}$ a* simple *length ensemble if* $\liminf_{i \to +\infty} \eta_i = 1$ *and* nonsimple *otherwise. We call $L_{\mathcal{X}}$ a* supersimple *or* unary-ended *length ensemble if* $\lim_{i \uparrow +\infty} \eta_i = 1$. *We call $L_{\mathcal{X}}$ a* pseudounary-ended *or* pseudounary *length ensemble if* $\lim_{i \uparrow +\infty} \eta_i$ *exists and is finite. We call $L_{\mathcal{X}}$ a* large *length ensemble if* $\limsup_{i \to +\infty} \eta_i = +\infty$ *and* nonlarge *otherwise. We call $L_{\mathcal{X}}$ a* superlarge *length ensemble if* $\lim_{i \uparrow +\infty} \eta_i = +\infty$.

For example, the trees in Figure 5.1 are nonsimple and nonlarge with $\lim_{i \uparrow +\infty} \eta_i = 2$. We may also use the above terms for the probability mass functions for which such codes are optimal, although a given probability mass function, if it has several optimal length ensembles, may be several of the above.

Figure 5.2 has two large examples which require some explanation. Consider the two-dimensional geometric distribution, $\boldsymbol{p}^{(\boldsymbol{2g_\pi})} = (1-\pi)^2 \cdot (1, \pi, \pi, \pi^2, \pi^2, \pi^2, \pi^3, \ldots)$. For any π, this would have a large tree, and we might expect this to be the easiest case in which one can find the optimal tree for a nondyadic, nonsimple, large probability mass function. However, no case has been solved for $\pi \in (0,1) \backslash \{\frac{1}{2}\}$. Tree (a) of Figure 5.2, related to the two-dimensional geometric distribution, is formed by starting with the two-element code and combining the root for the three-element code with the rightmost leaf node for the former (two-element) tree, and continuing on, combining the root of the nth tree — an $n+1$-element flat code tree — to a deepest root of the $n-1$th tree. This is a simple large tree, as $\liminf_{i \to +\infty} \eta_i = 1$ and $\limsup_{i \to +\infty} \eta_i = +\infty$. As $\pi \downarrow 0$, η for the optimal code tree for a two-dimensional

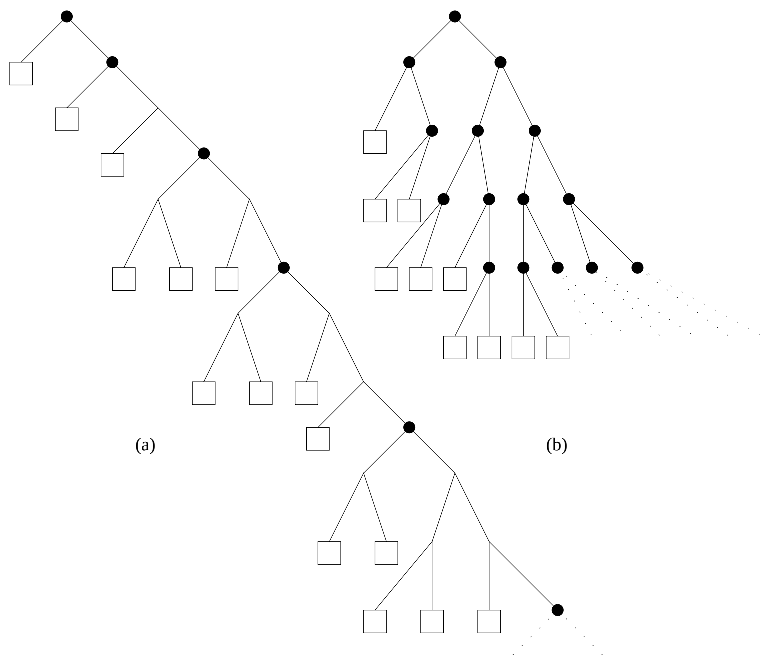


Figure 5.2: More examples of infinite coding trees

geometric distribution approaches η pointwise for this tree. Tree (b) of Figure 5.2 is the monotonic tree corresponding to $\eta_i = i$, a superlarge tree, since $\lim_{i \uparrow +\infty} \eta_i = +\infty$. This is the optimal tree for the two-dimensional geometric distribution at $\pi = \frac{1}{2}$.

One interesting question is whether or not tree (b) is optimal for any $\pi \neq \frac{1}{2}$, and, if so, for what range of values. This is an example of a question we may ask about infinite codes which sounds deceptively simple. Although we do not answer this, we consider the question as a means of presenting some techniques for analysis that may be fruitful for this and similar problems.

Note first that a tree is optimal if and only if the (strong) sibling property holds. (Recall that the strong sibling property states that, in an optimal tree, each nonroot node has a sibling and the nodes may be listed in order of nonincreasing weight, each node being adjacent in the list to its sibling.) Because of the structure of the tree, we need only check the nodes to the extreme left and right on each level to determine whether the code conforms to the sibling property. Thus it is easy to write an algorithm to empirically find the depth at which this property is violated for a given π. However, such an algorithm will not halt if this does not hold, so this does not solve our problem. An algebraic solution is also not apparent.

A different approach involves geometrically examining the set of probability mass functions for which a given tree is optimal. A given $L^*_{\mathcal{X}}$ is optimal iff $\forall\ L_{\mathcal{X}},\ \sum_i p_i l^*_i \leq \sum_i p_i l_i$, so the optimality regions are convex polytopes, due to these being intersections of half-hyperspaces. This view may be misleading, however, in a case with an infinite alphabet and thus infinite dimensions. For example, we might suspect that a parametrized continuous path converging on the "center point" — the corresponding dyadic distribution — would have an epsilon neighborhood in which the corresponding length ensemble is optimal. As a counterexample, consider the following path of probability vectors indexed by t:

$$\boldsymbol{q}(t) = (2^{-1}, 2^{-2}, \cdots, 2^{-\lfloor t \rfloor},\ (2 - \langle t \rangle)2^{-\lfloor t \rfloor - 1},\ \langle t \rangle 2^{-\lfloor t \rfloor - 1},\ 0,\ 0,\ \cdots) \tag{5.3}$$

This path converges to the dyadic geometric $\boldsymbol{p^{(dg)}} = (2^{-1}, 2^{-2}, \cdots)$ in metric space L^r — that in which distance is measured by $d(\boldsymbol{p}, \boldsymbol{q}) = \sum_i (p_i - q_i)^r$ — for all $r \geq 1$, the distance being bounded above by 2^{-t}. However, no point along this path has the same optimal solution as its dyadic limit point.

Most countably infinite alphabet distributions currently codable are variants of the geometric distribution, that is, for $\boldsymbol{p^{(g_\pi)}}$ of the form $p_i^{(g_\pi)} = (1-\pi)\pi^i$. For all π, the pure geometric distribution has a pseudounary (and thus nonlarge) optimal length ensemble. A pseudounary length ensemble is well-behaved in the sense that the value of η_i (and thus λ_i) eventually converges; thus the length ensemble corresponds to a code that may be represented as the concatenation of a finite alphabet code with, for some codewords, a unary code. All nondyadic nonsimple probability distributions with known optimal codes are pseudounary geometric variants [1, 28, 35].

Distributions other than geometric varieties include Poisson and Zeta distributions. The Poisson distribution with parameter $\lambda > 0$ has the form $p_i = i!^{-1}\lambda^i e^{-\lambda}$ for $i \geq 0$ and decays superexponentially. It is thus supersimple (unary-ended) and easy to characterize [44]. The Zeta distribution with parameter $\alpha > 1$ has the form

$p_i = \{[1+\zeta(\alpha)]i^\alpha\}^{-1}$ for $n \geq 1$ and decays subexponentially. ($\zeta(\alpha) \triangleq \sum_{k=1}^{+\infty} k^{-\alpha}$.) It is thus superlarge and has eluded solution.

Consider the special case of the inverse square distribution, $p_i = \frac{6}{\pi^2} i^{-2}$. This would seem to be the simplest instance of the Zeta distribution, and the first few lengths seem empirically derivable. However, neither the full solution nor this integer sequence has been characterized. Unlike geometric cases, the optimal solution to the Zeta distribution does not have a predictable structure, this due to the logarithmic relationship between probability and (ideal) length.

In fact, no superlarge optimal solution has been found for any nondyadic problem. The occurrence of the Zeta distribution is, however, far rarer than the occurrence of the geometric distribution and its variants, which come up in a number of predictive coding techniques, such as image coding [93].

The Golomb code, optimal for geometric probability distributions, is a pseudounary extension of the unary code [36]. For $p_0 < \Phi^{-1} = \frac{\sqrt{5}-1}{2}$, this is the unary code itself, a simple length ensemble. For $p_0 > \Phi^{-1}$, it is a nonsimple length ensemble. $p_0 = \Phi^{-1}$ has both simple and nonsimple optimal length ensembles [28, 35]. All are nonlarge.

If a probability mass function has a simple length ensemble as its optimal code and we can find an infinite subset of the values k for which $\eta_k = 1$, we can construct an arbitrarily large portion of the code tree. We briefly explain how to find and use this property. Let $p_j^s \triangleq \sum_{i=j+1}^{+\infty} p_i = 1 - \sum_{i=1}^{j} p_i$. Suppose there exist infinitely many j's in $\mathcal{X}$ such that $p_j \geq p_j^s$. It is immediate that for any optimal code with lengths l_i^*, $\eta_{l_j^*+1} = 1$ for these values of j [49].

Let us define $\mathcal{T}(j) \triangleq \{j+1, j+2, \ldots\}$ and the subset of $\boldsymbol{p}$ consisting of the corresponding weights $\boldsymbol{p}_{T(j)} \triangleq \cup_{i=j+1}^{+\infty}\{p_i\}$. (Recall that we use the term "weights" when we do not have $\sum_i p_i = 1$.) The optimal tree for these weights is thus a subtree of the overall optimal tree. (Recall that every subtree of an optimal tree is an optimal tree.) For any finite subset of the codewords $C_{\tilde{\mathcal{X}}} \subset C_{\mathcal{X}}$, we may thus build the finite tree for $\boldsymbol{p'} = (p_1, p_2, \cdots, p_j, p_j^s)$ — where j is the minimum $j \geq \max_{i \in \tilde{\mathcal{X}}} i$ such that $p_j \geq p_j^s$ — and calculate the desired codewords (or codeword lengths) from this.

Nonsimple length ensembles may also be calculated in some cases, although all known ones are program modeled cases; for example, if we know $\boldsymbol{p}$ is dyadic or geometric, we may use a Shannon or Golomb code, respectively, and this code may be nonsimple. In some sense, we expect most interesting probability functions to be nonsimple, as a simple distribution implies either a dramatic geometric decline or a lack of uniformity extending to infinity. However, as in the example illustrated by Figure 5.1, uniformity can lead to unknowable solutions. A general means of constructing the optimal average length code and/or proving its incalculability for finite entropy nonsimple probabilities has yet to be discovered.

This indicates that perhaps — with the exception of dyadic and simple distributions — we cannot code large distributions, or even nonpseudounary distributions. More investigation is needed to better understand this.

5.3 Extending to other penalties

5.3.1 Existence of optimal codes

The existence results of [64] may be carried over to quasilinear penalties. Consider continuous strictly monotonic $f : R^+ \to R^+$ (as proposed by Campbell) and $\boldsymbol{p} = (p_1, p_2, \ldots)$ such that

$$F^*(\boldsymbol{p}, f) \triangleq \inf_{\substack{\sum_i 2^{-l_i} \leq 1, \\ l_i \in \mathbb{Z}}} f^{-1} \left[\sum_{i=1}^{+\infty} p_i f(l_i) \right] \tag{5.4}$$

is finite. Consider, for an arbitrary $n \in \mathbb{N}$, optimizing for f with weights

$$\boldsymbol{p}^{(\boldsymbol{n})} \triangleq (p_1, p_2, \ldots, p_n, 0, 0, \ldots) \tag{5.5}$$

Denote an optimal code

$$C^{(n)} \triangleq \{c_1^{(n)}, c_2^{(n)}, \ldots, c_n^{(n)}, \Lambda, \Lambda, \ldots\} \tag{5.6}$$

where Λ is the null string and we use an infinite code — called a *binary truncated code* — for convenience. The codeword lengths of this binary truncated code are

$$L_{\mathcal{X}}^{(n)} \triangleq \{l_1^{(n)}, l_2^{(n)}, \ldots, l_n^{(n)}, 0, 0, \ldots\} \tag{5.7}$$

Thus, for convenience, $l_i^{(j)} = 0$ and $c_i^{(j)} = \Lambda$ for $i > j$. These lengths are also optimal for $(\sum_{j=1}^n p_j)^{-1} \cdot \boldsymbol{p}^{(n)}$, the vector of normalized weights.

Following [64], we say a sequence of binary sequence codes $C^{(1)}, C^{(2)}, C^{(3)}, \ldots$ *converges* to an infinite prefix-free code $C = \{c_1, c_2, \ldots\}$ if, for each i, the ith codeword in each code in the sequence is eventually codeword c_i.

Theorem 11 *Given Campbell f and $\mathbf{p}$ for which the corresponding $F^*(\boldsymbol{p}, f) < +\infty$, the following hold:*

1. *There exists a sequence of binary truncated codes that converges to an optimal code for $\mathbf{p}$; thus the infimum is achievable.*

2. *Any optimal binary code for $\mathbf{p}$ must satisfy the Kraft inequality with equality.*

Proof: Because here we are concerned only with cases in which the first length is at least 1, we may restrict ourselves to the domain $[f^{-1}[p_1 f(1)], +\infty)$, and may thus assume without loss of generality — as in Section 4.2 — that f is increasing. Recall

$$F^*(\boldsymbol{p}, f) = \inf_{\substack{\sum_i 2^{-l_i} \leq 1, \\ l_i \in \mathbb{Z}}} f^{-1}\left[\sum_{i=1}^{+\infty} p_i f(l_i)\right] < +\infty. \tag{5.8}$$

Then there exists an $L_{\mathcal{X}}' = \{l_1', l_2', l_3', \ldots\} \in \mathbb{Z}^{+\infty}$ such that

$$f^{-1}\left[\sum_{i=1}^{+\infty} p_i f(l_i')\right] < F^*(\boldsymbol{p}, f) + 1 \text{ and } \sum_{i=1}^{+\infty} 2^{-l_i'} \leq 1 \tag{5.9}$$

and thus

$$f^{-1}\left[\sum_{i=1}^{n} p_i f(l_i')\right] < F^*(\boldsymbol{p}, f) + 1 \text{ and } \sum_{i=1}^{n} 2^{-l_i'} < 1. \tag{5.10}$$

So, for minimizing $L_{\mathcal{X}}^{(n)}$, we have

$$f^{-1}\left[\sum_{i=1}^{n} p_i f(l_i^{(n)})\right] \leq f^{-1}\left[\sum_{i=1}^{n} p_i f(l_i')\right] < F^*(\boldsymbol{p}, f) + 1 \tag{5.11}$$

and

$$p_j f(l_j^{(n)}) \leq \sum_{i=1}^{n} p_i f(l_i^{(n)}) < f(F^*(\boldsymbol{p}, f) + 1) \tag{5.12}$$

for all j. This implies that

$$l_j^{(n)} < f^{-1}\left[\frac{f(F^*(\boldsymbol{p}, f) + 1)}{p_i}\right]. \tag{5.13}$$

Thus we have shown that, for any $i \in \mathbb{N}$, the sequence $l_i^{(1)}, l_i^{(2)}, l_i^{(3)}, \ldots$ is bounded. A corresponding sequence of codewords may therefore have only a finite set of values, so the sequence of codewords $c_i^{(1)}, c_i^{(2)}, c_i^{(3)}, \ldots$ — as well as any infinite subsequence — has a convergent subsequence. As in [64], we may conclude that there exists a subsequence of codes, $C^{(n_1)}, C^{(n_2)}, C^{(n_3)}, \ldots$, that converges to an infinite code $\widehat{C}$ with codeword lengths $\widehat{L_{\mathcal{X}}} = \{\widehat{l_1}, \widehat{l_2}, \widehat{l_3}, \ldots\}$. As a limit of prefix-free codes, this is itself a prefix-free code, and the codeword lengths converge pointwise to those of the convergent code.

We now show that $\widehat{C}$ is optimal. Let $\{\lambda_1, \lambda_2, \lambda_3 \ldots\}$ be the codeword lengths of an arbitrary prefix-free code. For every k, there is a $j \geq k$ such that $\widehat{l_i} = l^{(n_m)}$ for any $i \leq k$ if $m \geq j$. Due to the optimality of each $C^{(n)}$, for all $m \geq j$:

$$\begin{aligned}\sum_{i=1}^{k} p_i f(\widehat{l_i}) &= \sum_{i=1}^{k} p_i f(l_i^{(n_m)}) \qquad (5.14)\\ &\leq \sum_{i=1}^{n_m} p_i f(l_i^{(n_m)}) \leq \sum_{i=1}^{n_m} p_i f(\lambda_i). \qquad (5.15)\end{aligned}$$

Therefore,

$$\sum_{i=1}^{k} p_i f(\widehat{l_i}) \leq \sum_{i=1}^{n_m} p_i f(l_i^{(n_m)}) \leq \sum_{i=1}^{n_m} p_i f(\lambda_i) \leq \sum_{i=1}^{+\infty} p_i f(\lambda_i) \tag{5.16}$$

and, taking $k \to +\infty$, $\sum_i p_i f(\widehat{l_i}) \leq \sum_i p_i f(\lambda_i)$, leading directly to $f^{-1}\left[\sum_i p_i f(\widehat{l_i})\right] \leq f^{-1}\left[\sum_i p_i f(\lambda_i)\right]$ and the optimality of $\widehat{C}$.

Suppose the Kraft inequality is not satisfied with equality for optimal codeword lengths $\widehat{L_{\mathcal{X}}} = \{\widehat{l_1}, \widehat{l_2}, \ldots\}$. Then there is a $k \in \mathbb{N}$ such that $2^{-l_k} + \sum_i 2^{-l_i} < 1$. Consider code $\{\widehat{l_1}, \widehat{l_2}, \ldots, \widehat{l_{k-1}}, \widehat{l_k} - 1, \widehat{l_{k+1}}, \widehat{l_{k+2}}, \ldots\}$. This code satisfies the Kraft inequality and has penalty $f^{-1}[\sum_i p_i f(\widehat{l_i}) + \{f(\widehat{l_k} - 1) - f(\widehat{l_k})\} p_k)] < f^{-1}\left[\sum_i p_i f(\widehat{l_i})\right]$, and thus $\widehat{L_{\mathcal{X}}}$ is not optimal. Therefore the Kraft inequality must be satisfied with equality. ■

It may seem obvious that the Kraft inequality should be satisfied with equality for an optimal infinite code, but recall that an infinite full tree need not satisfy the Kraft inequality with equality. It is worthwhile at this point to examine the example in [64] of an infinite full tree failing equality.

Recall the technique of Elias described in Section 2.1, in which each codeword c_i of length l_i is viewed as occupying a portion of the $[0,1)$ interval beginning at its binary fractional expansion and occupying a subinterval of length 2^{-l_i}, as in Figure 2.1. The intervals are disjoint and that their summed length is the Kraft inequality sum.

Consider the following code C:

$$\begin{array}{lclllllllll}
c_1 & = & 0 & 0 & & & & & & & \\
c_2 & = & 0 & 1 & 0 & 0 & 0 & & & & \\
c_3 & = & 1 & 0 & 0 & 0 & 0 & & & & \\
c_4 & = & 1 & 1 & 0 & 0 & 0 & & & & \\
c_5 & = & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
c_6 & = & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & \vdots & & & &
\end{array} \tag{5.17}$$

This code has infinite number of codewords, each consisting of blocks of increasing size — two bits, then three bits, then four, etc. If a block consists of all zero bits, this indicates the codeword's termination. Otherwise, there is at least one more block in the codeword. Thus it is obviously a prefix-free code. However, we will show that, although corresponding to a full coding tree, it does not satisfy the Kraft inequality with equality.

Formalizing as in [64], let $\mathrm{CAT}(a_1, a_2)$ and the equivalent $\mathrm{CAT}_{i=1}^{2} a_i$ be two methods of denoting binary concatenation. Let N_j denote the set of j-bit binary strings excluding 0^j, the j-bit binary string consisting of all 0's. Thus the size of the set N_j is $|N_j| = 2^j - 1$. We may now formally define:

$$C = \{00\} \cup \left[\bigcup_{k=2}^{\infty} \mathrm{CAT} \left\{ \left(\mathop{\mathrm{CAT}}_{j=2}^{k} N_j \right), 0^{k+1} \right\} \right]. \tag{5.18}$$

If the tree were not full, there would exist a finite length sequence which is not the prefix of a valid codeword, does not contain a valid codeword as a prefix, and is not valid codeword itself, i.e., a sequence not compatible with the code. Suppose there exists such a sequence of length m. Note that we may pad the end of such a sequence with additional bits, because if the original sequence is incompatible, no compatible sequence contains it as a prefix. Let $k = \lceil \frac{1}{2}(-3 + \sqrt{9 + 8m}) \rceil$. This is the number of variable length blocks necessary to form the smallest codeword at least as long as this sequence, the first block being of length 2, the second of length 3, etc. We may pad the sequence with $[m - \frac{1}{2}(k+1)(k+2) - 1]$ zeroes, so it may be composed of such blocks.

Now we may show by induction that all possible sequences (of blocks) are compatible: It is obvious that one-block (two-bit) sequences are all valid codewords ($\{00\}$) or prefixes to valid codewords ($\{01000,10000,11000\}$). Assume all sequences with $k-1$ such block are prefixes for valid codewords (those for which the next k digits are 0), are valid codewords, or have valid codewords as prefixes. In the last two cases, this remains true for the k-block sequence. Thus we may assume that the first $k-1$ blocks comprise the prefix to a valid codeword. If the next k digits are 0, we have a valid codeword. Otherwise, we have a prefix to a valid codeword made by concatenating the (padded) sequence with 0^{k+1} ($k+1$ additional 0's). Thus no such incompatible sequence exists, and the tree is full.

Note that, although it is difficult to calculate the exact value of the Kraft inequality sum for this code, we can upper bound it as follows:

$$\begin{aligned}
\sum_i 2^{-l_i} &= \frac{1}{4} + \sum_{k=2}^{\infty} \left| \mathrm{CAT} \left\{ \left(\underset{j=2}{\overset{k}{\mathrm{CAT}}}\, N_j \right), 0^{k+1} \right\} \right| \left(2^{-\sum_{n=2}^{k+1} n} \right) && (5.19) \\
&= \frac{1}{4} + \sum_{k=2}^{\infty} \left(\prod_{j=2}^{k} (2^j - 1) \right) \left(2^{-\sum_{n=2}^{k+1} n} \right) && (5.20) \\
&< \frac{1}{4} + \sum_{k=2}^{\infty} \left(2^{\sum_{n=2}^{k} n} \right) \left(2^{-\sum_{n=2}^{k+1} n} \right) && (5.21) \\
&= \sum_{k=1}^{\infty} 2^{-(k+1)} && (5.22) \\
&= \frac{1}{2} && (5.23)
\end{aligned}$$

Equation (5.19) finds an expression for the Kraft inequality sum by multiplying the number of codewords of each possible length (length being equal to $\sum_{n=2}^{k+1} n$ for a k-block codeword) by their Kraft inequality term ($2^{-l_i} = 2^{-\sum_{n=2}^{k+1} n}$). Equation (5.20) calculates explicitly the number of codewords of each length. Equation (5.21) increases each multiplicative term to the smaller power of two greater than or equal to it. Equation (5.22) cancels redundant terms and equation (5.23) calculates an infinite series.

The Kraft inequality is not satisfied with equality because each valid codeword of length m is required to end in $\frac{1}{2}(-1+\sqrt{9+8m})$ zeroes. Thus, although full, this code may be considered wasteful. Note that we may also use the analogy of Figure 2.1 to explain this. Because we have an infinite number of codewords, it is possible to fill in the $[0, 1)$ interval in such a way as to leave no gaps and yet have the filling intervals sum to a number less than 1.

We thus have yet another example of how infinite coding differs from finite coding, a reminder that we should not assume properties carry over, just as we should not assume properties from the linear penalty carry over to other penalties. Next, however, we see that one basic property — having to do with entropy — does.

Theorem 12 *Recall that*

$$H(\boldsymbol{p}, f) \triangleq \inf_{\substack{\sum_i 2^{-l_i} \leq 1, \\ l_i \in \mathbb{R}}} f^{-1}\left[\sum_i p_i f(l_i)\right] \tag{5.24}$$

for $f : \mathbb{R}^+ \rightarrow \mathbb{R}^+$. If $H(\boldsymbol{p}, f) < +\infty$ and either f is subtranslatory (see Section 4.2) or $f(x+1) = O(f(x))$ (which includes all concave cases and all decreasing cases), then the coding problem

$$F^*(\boldsymbol{p}, f) = \inf_{\substack{\sum_i 2^{-l_i} \leq 1, \\ l_i \in \mathbb{Z}}} f^{-1}\left[\sum_i p_i f(l_i)\right] \tag{5.25}$$

has a minimizing $L^*_{\mathcal{X}}$ resulting in a finite value for $F^*(\boldsymbol{p}, f) = F(L^*_{\mathcal{X}}, \boldsymbol{p})$.

Proof: Assume without loss of generality that f is increasing. If f is subtranslatory, then $F^*(\boldsymbol{p}, f) < 1 + H(\boldsymbol{p}, F) < +\infty$. If $f(x+1) = O(f(x))$, then there are $\alpha, \beta > 0$ such that $f(x+1) \leq \max\{\alpha, \beta f(x)\}$ for all x. Then

$$\begin{aligned} f^{-1}\left[\sum_i p_i f(l_i + 1)\right] &\leq f^{-1}\left[\sum_i p_i \max\{\alpha, \beta f(l_i)\}\right] && (5.26) \\ &< f^{-1}\left[\sum_i p_i \{\alpha + \beta f(l_i)\}\right] && (5.27) \\ &= f^{-1}\left[\alpha + \beta \sum_i p_i f(l_i)\right] && (5.28) \end{aligned}$$

So $F^*(\boldsymbol{p}, f) < f^{-1}\left[\alpha + \beta f\left(H(\boldsymbol{p}, f)\right)\right] < +\infty$, and the infimum, which we already proved may be minimized, is finite. ■

5.3.2 Construction of optimal codes

The constructive results of Kato, Han, and Nagaoka in [49] may also be extended, albeit only to the exponential penalty. Also, because the linear case is the only case in which weights strictly sum, the results are not as strong. Nevertheless, a generalization may be made.

Suppose we know that an optimal exponential penalty tree for a given probability mass function $\boldsymbol{w}$ is a simple tree. Then for this optimal code of lengths $L^*_{\mathcal{X}}$, $\eta_{l^*_j+1} = 1$ for an infinite number of values of j. In other words, for these values of j, if $\mathcal{T}(j) \triangleq \{j+1, j+2, \ldots\}$, an optimal tree for the subset of $\boldsymbol{w}$, $\boldsymbol{w}_{\mathcal{T}(j)}$, is a subtree of the optimal tree.

We can use equation (2.22) to see that, on each subtree associated with such a value of j,

$$\begin{aligned} \left[\sum_i \left(\frac{w_i}{\sum_k w_k}\right)^{\frac{1}{1+b}}\right]^{\frac{1+b}{b}} &\leq \left[\sum_i \left(\frac{w_i}{\sum_k w_k}\right) 2^{bl^*_i}\right]^{\frac{1}{b}} && (5.29) \\ &< 2\left[\sum_i \left(\frac{w_i}{\sum_k w_k}\right)^{\frac{1}{1+b}}\right]^{\frac{1+b}{b}} && (5.30) \end{aligned}$$

where all summations are over $\mathcal{T}(j)$. This may be simplified and divided into three cases as follows:

$$\begin{array}{rcccll} (\sum_{i=j+1}^{+\infty} w_i^{\frac{1}{1+b}})^{1+b} & \leq & w^s_j & < & 2^b(\sum_{i=j+1}^{+\infty} w_i^{\frac{1}{1+b}})^{1+b}, & b > 0 \\ & & w^s_j & = & \sum_{i=j+1}^{+\infty} w_i, & b = 0 \\ 2^b(\sum_{i=j+1}^{+\infty} w_i^{\frac{1}{1+b}})^{1+b} & < & w^s_j & \leq & (\sum_{i=j+1}^{+\infty} w_i^{\frac{1}{1+b}})^{1+b}, & b < 0 \end{array} \quad (5.31)$$

where $w^s_j \triangleq \sum_{i=j+1}^{+\infty} w_i 2^{bl^*_i}$ and l^*_i's are lengths for an optimal infinite alphabet code. Since indices $\mathcal{T}(j)$ correspond to weights forming an optimal subtree within the optimal tree, the weight on the subtree root node is w^s_j. From the above equations, we may derive

$$w^{j,b}_{max} \triangleq 2^{(b^+)} \left(\sum_i w_i^{\frac{1}{1+b}}\right)^{1+b} \quad (5.32)$$

where $b^+ \triangleq \max(0, b)$ and

$$w_{min}^{j,b} \triangleq 2^{(b^-)} \left(\sum_i w_i^{\frac{1}{1+b}} \right)^{1+b} \tag{5.33}$$

where $b^- \triangleq \min(0, b)$, so that $w_j^s \in [w_{min}^{j,b}, w_{max}^{j,b}]$.

Using this, we may be able to build (an arbitrarily large subset of) an optimal exponential Huffman code in some cases where there exist infinitely many j's in $\mathcal{X}$ such that $w_j \geq w_{max}^{j,b}$. For any finite subset of codewords $C_{\tilde{\mathcal{X}}} \subset C_{\mathcal{X}}$, we can find the minimum $j \geq \max_{i \in \tilde{\mathcal{X}}} i$ such that $w_j \geq w_{max}^{j,b}$, and build a finite exponential tree for $w_{max}^{[1,j]} \triangleq (w_1, w_2, \cdots, w_j, w_{max}^{j,b})$ as well as one for $w_{min}^{[1,j]} \triangleq (w_1, w_2, \cdots, w_j, w_{min}^{j,b})$. If these result in the same length ensemble, this is the length ensemble to use. Otherwise, we could try the next j satisfying the aforementioned criterion, continuing to try such values until we find the codeword lengths or until some cut-off point at which we would give up.

Note that we are not guaranteed that such an algorithm will work unless $b = 0$. (Values close to 0, however, are more likely to work than those not as close.) Also note that, although w_j^s could be any value in $[w_{min}^{j,b}, w_{max}^{j,b}]$, because all $(w_1 \; w_2 \; \cdots \; w_j \; w_j^s)$ are linear combinations of $w_{max}^{[1,j]}$ and $w_{min}^{[1,j]}$, testing only the two boundary values is sufficient.

For best results, bottom-merging [84] should be used when building a tree using $w_{min}^{[1,j]}$ and top-merging, preferring merged items to singletons, when building a tree using $w_{max}^{[1,j]}$, so that, if there are multiple solutions to either case and two optimal trees common to the two cases, the same tree is chosen.

In addition, only in the case of $b = 0$ can we apply this method directly in an oracle modeled case, as we need to calculate an infinite sum. However, we can in practice usually find such infinite sums, and, failing that, may be able to bound $[w_{min}^{j,b}, w_{max}^{j,b}]$.

Chapter 6

Concluding Remarks

In this chapter, we briefly summarize the main contributions of this dissertation and discuss promising directions for further research.

6.1 Summary of contributions

The four main contributions of this dissertation are:

1. Extensions of Huffman coding to variants of the exponential penalty with deterministic solutions, to the siege problem, to minimal maximal pointwise redundancy, and to minimal d-average b-redundancy, the last of which includes several previously examined penalties;

2. A method for solving the generalized quasilinear convex coding problem, which includes, may be extended to, and improves upon earlier penalties;

3. Approaches to infinite alphabet coding for nontrivial nonlinear penalties;

4. Coding bounds, frameworks, and other properties for these and other general problems with alternative coding penalties.

Figure 1.2 presents a visual summary of finite alphabet cases, with examples of solved problems given in Section 4.9.

6.2 Extensions and future directions

The previous chapter lays down a framework for seeing whether codes on infinite alphabets are existent [35, 64] or calculable [28, 36, 49, 71] for certain instances and/or distributions of such problems, both for the standard linear penalty and for other penalties. Further bounds, limits, and heuristics would also be interesting. For example, how does the sequence of solutions for a limiting sequence of probabilities behave with respect to the optimal solution? This has been indirectly explored previously in [19, 64].

It would be interesting to see if the above methods might generalize to other families of penalties, such as a general family of nonquasilinear or nonexponential nonconvex penalties. An example of a nonexponential nonconvex penalty is the geometric mean, given as F_{16} in Table 1.1, which may be presented in the form of equation (2.3) with $f(x) = \log x$. Nonquasilinear objectives are also of interest. A trivial example of such an objective is minimizing median length, which may be done by finding i_{med}, the minimum i for which $\sum_{j=1}^{i} p_j > \frac{1}{2}$, and assigning codeword lengths $\lceil \lg(i_{med}+1) \rceil$ to the i_{med} most likely items and arbitrary codeword lengths satisfying the Kraft inequality to the rest. We discussed the delay penalty, given in equation (4.45), and possibilities for generalization; other such practical yet complicated penalties might also be soluble.

The situation of maximizing the probability of some measure of success — considered first by Humblet in [45] and also by us in Section 2.4.1 — is one worth further examination, particularly with the rising importance of communication over unreliable networks. For example, processes with memory in problems similar to that of Section 2.4.1 could be examined. Along the same lines, because we do not always have a lossless channel in real-world situations, formulation and solution of low bitrate source-channel coding problems would also be of interest.

Competitive optimality [22] is another criterion, but as there exists no competitively optimal code for many probabilities, this does not have the generality of the other penalties considered. Furthermore, when it does exist, it is an optimal solution to standard (Huffman) coding [97].

On a related note, it may be good to explicitly prove and further examine the extensions to Lemmas 2–4 and Theorems 5–6 alluded to in Section 2.5.3. We also might find several other implications of bottom-merge and top-merge coding, as in the linear case [27, 55, 68], especially for more general convex penalties.

The problem space may be expanded by imposing constraints on codewords — on alphabetic order [42], on runlength and/or charge [67], on suffixes and/or prefixes [10,17,98]. For example, generalizations of the Hu-Tucker algorithm for alphabetically ordered trees, as in [40, 47, 61], should be similar to those of the Huffman algorithm.

If we do not know the exact probability distribution, but rather a class of probability distributions, we may want to use universal techniques such as those used for the linear penalty in [21] and for minimal maximal pointwise redundancy in [24]. These results may be generalizable.

Stricter upper bounds on DABR may exist as in the linear case; such linear case upper bounds may be found in [29,98], among others. Note that the bound for maximum b-redundancy is trivial. (A minimax redundancy code on a uniform alphabet of size 2^n+1 has maximum redundancy $r_{max} = n+1-\lg(2^n+1)$, which approaches 1, a supremum previously established.) Because linear coding is the only case in which a subtree of weights is always the weight of the weight sum, these bounds may be elusive.

Finally, although many of the above results are easily extendable to nonbinary coding (that with ≥ 3 output symbols), other results, such as those of the generalized quasilinear convex coding algorithm, may not be. Thus it may be worthwhile to find such extensions.

Bibliography

[1] J. Abrahams, "Huffman-type Codes for Infinite Source Distributions," *Journal of the Franklin Institute*, Vol. 331B(3), pp. 265-271, 1994; also in *Proceedings, IEEE Data Compression Conference*, pp. 83-89, Mar. 29-31, 1994.

[2] J. Abrahams, "Huffman Code Trees and Variants," *DIMACS Workshop on Codes and Trees: Algorithmic and Information Theoretic Approaches*, Rutgers University, Piscataway, NJ, 1998.

[3] J. Abrahams, "Code and parse trees for lossless source encoding," *Communications in Information and Systems*, Vol. 1, pp. 113-146, 2001.

[4] J. Aczél, "On Mean Values," *Bulletin of the American Mathematical Society*, Vol. 54, pp. 392-400, 1948.

[5] J. Aczél, "Determination of All Additive Quasiarithmetic Mean Codeword Lengths," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, Vol. 29, pp. 351-360, 1974.

[6] J. Aczél, "On Shannon's Inequality, Optimal Coding, and Characterizations of Shannon's and Rényi's Entropies," *Symposia Mathematica*, Istituto Nazionale di Alta Matematica, Roma, 1973, Vol. 15, pp. 153-179, Academic Press, New York, NY, 1975.

[7] J. Aczél and Z. Daróczy, *On Measures of Information and Their Characterizations*, Academic Press, New York, NY, 1975.

[8] J. Aczél and J. Dhombres, *Functional Equations in Several Variables*, Cambridge University Press, Cambridge, MA, 1989.

[9] A. Aggerwal, B. Schieber, and T. Tokuyama, "Finding a minimum-weight k-link path on graphs with the concave Monge property and applications," *Discrete and Computational Geometry*, Vol. 12, pp. 263-280, 1994.

[10] T. Berger and R. W. Yeung, "Optimum '1' ended binary prefix codes," *IEEE Transactions on Information Theory*, Vol. IT-36, pp. 1435-1441, 1990.

[11] A. Bookstein and S. T. Klein, "Is Huffman Coding Dead?" *Computing*, Vol. 50, pp. 279-296, 1993.

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*, preprint, 2002 [Online, http://www.stanford.edu/˜boyd/cvxbook.html, retrieved May 2003].

[13] M. Buro, "On the maximum length of Huffman codes," *Information Processing Letters*, Vol. 31, pp. 219-234, 1993.

[14] L. L. Campbell, "A Coding Problem and Rényi's Entropy," *Information and Control*, Vol. 8, pp. 423-429, 1965.

[15] L. L. Campbell, "Definition of Entropy by Means of a Coding Problem," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, Vol. 6, pp. 113-118, 1966.

[16] R. M. Capocelli, A. de Santis, and G. Persiano, "Binary prefix codes ending in a '1'," *IEEE Transactions on Information Theory*, Vol. IT-40, pp. 1296-1302, 1994.

[17] S. Chan and M. J. Golin, "A dynamic programming algorithm for constructing optimal '1'-ended binary prefix-free codes," *IEEE Transactions on Information Theory*, Vol. IT-46, pp. 1637-1644, 2000.

[18] C. Chang and J. Thomas, "Huffman algebras for independent random variables," *Discrete Event Dynamic Systems*, Vol. 4, pp. 23-40, 1994.

[19] T. Chow and M. Golin, "Convergence and Construction of Minimal-Cost Infinite Trees," Manuscript based on talk at *International Symposium on Information Theory*, Lausanne, Switzerland, 1999.

[20] D. Cohen and M. L. Friedman, "Weighted Binary Trees for Concurrent Searching," *SIAM Journal of Algorithms*, Vol. 20, pp. 87-112, 1996.

[21] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley-Interscience, New York, NY, 1991.

[22] T. M. Cover, "On the Competitive Optimality of Huffman Codes," *IEEE Transactions on Information Theory*, Vol. IT-37, pp. 172-174, 1991.

[23] G. Dial, "On a Coding Theorem Connected with Entropy of Order-Alpha and Type-Beta," *Information Sciences*, Vol. 30, pp. 55-65, 1983.

[24] M. Drmota and W. Szpankowski, "Generalized Shannon Code Minimizes the Maximal Redundancy," *Proceedings, Latin American Theoretical Informatics (LATIN)*, Cancun, Mexico, 2002.

[25] R. Durrett, *Probability: Theory and Examples*, Duxbury Press, Belmont, CA, 1996.

[26] C. Flores, *Encoding of Bursty Sources Under a Delay Criterion*, Ph.D. Thesis, University of California, Berkeley, 1983.

[27] G. Forst and A. Thorup, "Minimal Huffman trees," *Acta Informatica*, Vol. 36, pp. 721-734, 2000.

[28] R. G. Gallager and D. C. Van Voorhis, "Optimal Source Codes for Geometrically Distributed Integer Alphabets," *IEEE Transactions on Information Theory*, Vol. IT-21, pp. 228-230, 1975.

[29] R. G. Gallager, "Variations on a Theme by Huffman," *IEEE Transactions on Information Theory*, Vol. IT-24, 1978.

[30] R. G. Gallager, *Discrete Stochastic Processes*, Kluwer Academic Publishers, Boston, MA, 1996.

[31] M. R. Garey, "Optimal binary search trees with restricted maximal depth," *SIAM Journal on Computing*, Vol. 3, pp. 101-110, 1974.

[32] M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Company, San Francisco, CA, 1979.

[33] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, MA, 1992.

[34] C. R. Glassy and R. M. Karp, "On the Optimality of Huffman Trees," *SIAM Journal on Applied Mathematics*, Vol. 31, pp. 368-378, 1976.

[35] M. J. Golin, "A Personal View of Generalized Huffman Encoding," *Proceedings, Combinatorics of Searching Sorting and Coding*, Ischia Island, Italy, 2001.

[36] S. W. Golomb, "Run-length encodings," *IEEE Transactions on Information Theory*, Vol. IT-12, pp. 399-401, 1966.

[37] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 1991.

[38] Y. Horibe, "Remarks on 'compact' Huffman trees," *Journal of Combinatorics, Information and System Sciences*, Vol. 9, pp. 117-120, 1984.

[39] T. C. Hu, D. J. Kleitman, and J. K. Tamaki, "Binary Trees Optimum Under Various Criteria," *SIAM Journal on Applied Mathematics*, Vol. 37, pp. 246-256, 1979.

[40] T. C. Hu and M. T. Shing, *Combinatorial Algorithms*, Second edition, Dover Publications, Mineola, NY, 2002.

[41] T. C. Hu and K. C. Tan, "Path length of binary search trees," *SIAM Journal on Applied Mathematics*, Vol. 22, pp. 225-234, 1972.

[42] T. C. Hu and A. C. Tucker, "Optimal computer search trees and variable length alphabetic codes," *SIAM Journal on Applied Mathematics*, Vol. 21, pp. 514-532, 1971.

[43] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, Vol. 40, pp. 1098-1101, 1952.

[44] P. A. Humblet, "Optimal Source Coding for a Class of Integer Alphabets," *IEEE Transactions on Information Theory*, Vol. IT-24, pp. 110-112, 1978.

[45] P. A. Humblet, "Generalization of Huffman Coding to Minimize the Probability of Buffer Overflow," *IEEE Transactions on Information Theory*, Vol. IT-27, pp. 230-232, 1981.

[46] F. K. Hwang, "On finding a single defective in binomial group testing," *Journal of American Statistical Association*, Vol. 69, pp. 146-150, 1974.

[47] M. Karpinski, L. L. Larmore, and W. Rytter, "Correctness of constructing optimal alphabetic trees revisited," *Theoretical Computer Science*, Vol. 180, pp. 309-324, 1997.

[48] J. Katajainen, A. Moffat, and A. Turpin, "A Fast and Space-Economical Algorithm for Length-Limited Coding," *Proceedings of the International Symposium on Algorithms and Computation*, Cairns, Australia, p. 1221, Dec. 1995.

[49] A. Kato, T. S. Han, and H. Nagaoka, "Huffman Coding with an Infinite Alphabet," *IEEE Transactions on Information Theory*, Vol. IT-42, pp. 977-984, 1996.

[50] G. O. H. Katona and T. O. H. Nemetz, "Huffman Codes and Self-Information," *IEEE Transactions on Information Theory*, Vol. IT-22, pp. 337-340, 1976.

[51] A. Y. Khinchin, "Математическая теория стационарнойочереди" ("Mathematical Theory of Stationary Queues"), Matematicheskii Sbornik, Vol. 39, pp. 73-84, 1932.

[52] J. F. C. Kingman, "Inequalities in the Theory of Queues," *Journal of the Royal Statistical Society*, Ser. B, Vol. 32, pp. 102-110, 1970.

[53] D. E. Knuth, "Huffman's Algorithm via Algebra," *Journal of Combinatorial Theory*, Ser. A, Vol. 32, pp. 216-224, 1982.

[54] D. E. Knuth, *The Art of Computer Programming, Volume I/Fundamental Algorithms*, Third edition, Addison-Wesley, Reading, MA, 1997.

[55] L. T. Kou, "Minimum Variance Huffman Codes," *SIAM Journal on Computing*, Vol. 9, pp. 138-148, 1982; original as *Minimal Variance Huffman Codes*, Research report RC 8333, International Business Machines Corporation, 1980.

[56] S. Kumar and M. Sobel, "Finding a single defective in binomial group testing," *Journal of American Statistical Association*, Vol. 66, pp. 824-828, 1971.

[57] L. L. Larmore, "Height restricted optimal binary trees," *SIAM Journal on Computing*, Vol. 16, pp. 1115-1123, 1987; original as ICS TR-86-03, Department of Information and Computer Science, University of California, Irvine, February 1986.

[58] L. L. Larmore, "Minimum Delay Codes," *SIAM Journal on Computing*, Vol. 18, pp. 82-94, 1989.

[59] L. L. Larmore and D. S. Hirschberg, "A Fast Algorithm for Optimal Length-Limited Huffman Codes," *Journal of the Association for Computing Machinery*, Vol. 37, pp. 464-473, 1990.

[60] L. L. Larmore and T. M. Przytycka, "Parallel Construction of Trees With Optimal Weighted Path Length," *Proceedings of the Third ACM Symposium on Parallel Algorithms and Architectures*, pp. 71-80, 1991.

[61] L. L. Larmore and T. M. Przytycka, "A Fast Algorithm for Optimum Height-Limited Alphabetic Binary-Trees," *SIAM Journal on Computing*, Vol. 23, pp. 1283-1312, 1994.

[62] L. L. Larmore and T. M. Przytycka, "A Parallel algorithm for optimum height limited alphabetic binary trees," *Journal of Parallel and Distributed Computing*, Vol. 35, pp. 49-56, 1996.

[63] M. Liddell and A. Moffat, "Incremental Calculation of Optimal Length-Restricted Codes," *Proceedings, IEEE Data Compression Conference*, Snowbird, Utah, pp. 182-191, Apr. 2002.

[64] T. Linder, V. Tarokh, and K. Zeger, "Existence of Optimal Prefix Codes for Infinite Source Alphabets," *IEEE Transactions on Information Theory*, Vol. IT-43, pp. 2026-2028, 1997.

[65] G. Longo and G. Galasso, "An application of informational divergence to Huffman codes," *IEEE Transactions on Information Theory*, Vol. IT-28, pp. 36-43, 1982.

[66] U. Manber, *Introduction to Algorithms*, Addison-Wesley, Reading, MA, 1989.

[67] B. Marcus, P. Siegel, and R. Roth, *An Introduction to Coding for Constrained Systems*, 2000 [Online, http://www.stanford.edu/class/ee392p/, retrieved May 2003]; original in W. C. Huffman and V. Pless, ed., *Handbook of Coding Theory*, Chapter 20, Elsevier Press, New York, NY, 1998.

[68] G. Markowsky, "Best Huffman Trees," *Acta Informatica*, Vol. 16, pp. 363-370, 1981.

[69] B. McMillan, "Two inequalities implied by unique decipherability," *IRE Transactions on Information Theory*, Vol. IT-2, pp. 115-116, 1956.

[70] P. Mendenhall, "Cell phones were rebels' downfall." *MSNBC News*, Oct. 26, 2002 [Online, http://www.msnbc.com/news/826347.asp, retrieved November 2002].

[71] N. Merhav, G. Seroussi, and M. Weinberger, "Optimal Prefix Codes for Sources with Two-Sided Geometric Distributions," *IEEE Transactions on Information Theory*, Vol. IT-46, pp. 121-135, 2000.

[72] R. L. Milidiú and E. S. Laber, "The WARM-UP Algorithm: A Lagrangian Construction of Length Restricted Huffman Codes," *SIAM Journal on Computing*, Vol. 30, pp. 1405-1426, 2000.

[73] H. Minc, "A problem in partitions: Enumeration of elements of a given degree in the free commutative entropic cyclic groupoid," *Proceedings of the Edinburgh Mathematical Society*, Ser. 2, Vol. 11, pp. 223-224, 1959.

[74] A. Moffat, A. Turpin, and J. Katajainen, "Space-Efficient Construction of Optimal Prefix Codes," *Proceedings, IEEE Data Compression Conference*, Snowbird, Utah, pp. 192-202, Mar. 28-30, 1995.

[75] H. Murakami, S. Matsumoto, and H. Yamamoto, "Algorithm for Construction of Variable Length Code with Limited Maximum Word Length," *IEEE Transactions on Communications*, Vol. COM-32, 1157-1159, 1984.

[76] P. Nath, "On a coding theorem connected with Rényi entropy," *Information and Control*, Vol. 29, pp. 234-242, 1975.

[77] E. Norwood, "The Number of Different Possible Compact Codes," *IEEE Transactions on Information Theory*, Vol. IT-13, pp. 613-616, 1967.

[78] D. S. Parker, Jr., "Optimality of the Huffman Algorithm," *SIAM Journal on Computing*, Vol. 9, pp. 470-489, 1980; erratum, Vol. 27, p. 317, 1998.

[79] F. Pollaczek, "Über eine Aufgabe der Wahrscheinlichkeitstheorie, I-II," *Mathematische Zeitschrift*, Vol. 32, pp. 64-100 and 729-750, 1930.

[80] A. Rényi, "Some Fundamental Questions of Information Theory," *Magyar Tudományos Akadémia III. Osztalyanak Közlemenyei*, pp. 251-282, 1960.

[81] A. Rényi, "On Measures of Entropy and Information," *Proceedings of 4th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 547-561, 1960.

[82] A. Rényi, *Naplò az információelméletről (A Diary on Information Theory)*, Gondolat, Budapest, Hungary, 1969.

[83] B. Schieber, "Computing a minimum-weight k-link path in graphs with the concave Monge property," *Journal of Algorithms*, Vol. 29, pp. 204-222, 1998.

[84] E. S. Schwartz, "An Optimum Encoding with Minimum Longest Code and Total Number of Digits," *Information and Control*, Vol. 7, pp. 37-44, 1964.

[85] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, Vol. 27, pp. 379-423, 1948.

[86] A. Shenhar, *A Theory of Intrinsic Information and Computation on Topological Spaces*, Ph.D. Thesis, Stanford University, 1976.

[87] J. K. Tamaki, *Optimal Binary Trees and Sequences Realized by Eulerian Triangulations*, Ph.D. Thesis, Massachusetts Institute of Technology, 1978.

[88] A. Turpin and A. Moffat, "Practical Length-limited Coding for Large Alphabets," *The Computer Journal*, Vol. 38, pp. 339-347, 1995.

[89] A. Turpin and A. Moffat, "Efficient Implementation of the Package-Merge Paradigm for Generating Length-Limited Codes," *Proceedings of Computing: The Australasian Theory Symposium*, Melbourne, Australia, pp. 187-195, January 29-30, 1996.

[90] P. van Emde Boas, *Another NP-Complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice*, Technical Report #81-04, Mathematical Institute, University of Amsterdam, Amsterdam, Netherlands, 1981.

[91] J. van Leeuwen, "On the construction of Huffman trees," *Proceedings 3rd International Colloquium on Automata, Languages, and Programming*, University of Edinburgh, Edinburgh, Scotland, pp. 382-410, 1976.

[92] D. C. Van Voorhis, "Constructing codes with bounded codeword lengths," *IEEE Transactions on Information Theory*, Vol. IT-20, pp. 288-290, 1974.

[93] M. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS," *IEEE*

Transactions Image Processing, Vol. 9, pp. 1309-1324, 2000; original as Hewlett-Packard Laboratories Technical Report No. HPL-98-193R1, November 1998, revised October 1999.

[94] P. Whittle, *Risk-sensitive Optimal Control*, John Wiley & Sons, Chichester, West Sussex, UK, 1990.

[95] I. H. Witten, A. Moffat, and T. Bell, *Managing Gigabytes*, Second edition, Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[96] A. D. Wyner, "On the Probability of Buffer Overflow Under an Arbitrary Bounded Input-Output Distribution," *SIAM Journal on Applied Mathematics*, Vol. 27, pp. 544-570, 1974.

[97] H. Yamamoto and T. Itoh, "Competitive Optimality of Source Codes," *IEEE Transactions on Information Theory*, Vol. IT-41, pp. 2015-2019, 1995.

[98] C. Ye and R. W. Yeung, "Some Basic Properties of Fix-Free Codes," *IEEE Transactions on Information Theory*, Vol. IT-47, pp. 72-87, 2001.

[99] C. Ye and R. W. Yeung, "Redundancy of Huffman codes," *IEEE Transactions on Information Theory*, Vol. IT-48, pp. 2132-2138, 2002.