RANDOM NUMBER GENERATION USING A BIASED SOURCE

Sung-il Pae, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 2005
Professor Michael C. Loui, Advisor

We study random number generation using a biased source motivated by previous works on this topic, mainly, von Neumman (1951), Elias (1972), Knuth and Yao (1976) and Peres (1992). We study the problem in two cases: first, when the source distribution is *unknown*, and second, when the source distribution is *known*. In the first case, we characterize the functions that use a discrete random source of unknown distribution to simulate a target discrete random variable with a given rational distribution. We identify the functions that minimize the ratio of source inputs to target outputs. We show that these optimal functions are efficiently computable. In the second case, we prove that it is impossible to construct an optimal tree algorithm recursively, using algebraic decision procedures. Our model of computation is sufficiently general to encompass previously known algorithms for this problem.

RANDOM NUMBER GENERATION USING A BIASED SOURCE

BY

SUNG-IL PAE

B.S., Seoul National University, 1993
M.S., University of Illinois at Urbana-Champaign, 1997

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2005

Urbana, Illinois

ABSTRACT

We study random number generation using a biased source motivated by previous works on this topic, mainly, von Neumman (1951), Elias (1972), Knuth and Yao (1976) and Peres (1992). We study the problem in two cases: first, when the source distribution is *unknown*, and second, when the source distribution is *known*. In the first case, we characterize the functions that use a discrete random source of unknown distribution to simulate a target discrete random variable with a given rational distribution. We identify the functions that minimize the ratio of source inputs to target outputs. We show that these optimal functions are efficiently computable. In the second case, we prove that it is impossible to construct an optimal tree algorithm recursively, using algebraic decision procedures. Our model of computation is sufficiently general to encompass previously known algorithms for this problem.

ACKNOWLEDGMENTS

I am deeply grateful to my advisor Michael C. Loui. He provided me with advice on almost every aspect of my life as a graduate student, not to mention the countless meetings and comments on my thesis work. I began working with him at an especially difficult time of my graduate study. Without his encouragements and guidance, the complexity of my life would have been even greater.

I would like to express my gratitude to members of my doctoral committee, Jeff Erickson, Jack Lutz, Lenny Pitt and Andrew Singer. They suggested many improvements on my thesis and provided me with ideas to work on further beyond the thesis.

I would like to thank Jean Ponce for his support during my first years of graduate study and his guidance.

Office of Information Management provided me with assistantship for many years. I would like to thank Nursalim Hadi and people at OIM for enduring me and for what I learned while I worked for OIM.

During my long stay in Urbana, I became aquainted with many nice people. Particularly, I would like to thank Donald and Jean Burkholder, Robert Kaufmann, Jang-Mei Wu, Robert and Norma McKim for their kindness.

My thanks go to many colleagues of mine for the fun time spent together, suggestions, advices and discussions that sometimes helped my research. I would like to thank especially Jayhoon Byun, Jin-Yeop Chang, Yakup Genc, Kyungwon Hwang, Eunhee Kim,

Hongseok Kim, Hwangnam Kim, Youngihm Kho, Youngmin Kwon, Kihwal Lee, Jaejin Lee, Mike Roman, Attawith Sudsang, Hongseok Yang and Seung Yi.

I would like to express my deep gratitude for the unconditional love, support and patience of my parents and parent-in-laws in Korea. My love and thanks to my wife, Jiyeon, for all the happiness and the unhappiness we shared together. All my love to my daughter, Jiheh, whose timely arrival in our life made us better human beings.

TABLE OF CONTENTS

CHAPTER **PAGE**

CHAPTER 1

Introduction

Anyone who considers arithmetical methods of producing random digits is, of course, living in a state of sin.

— VON NEUMANN (1951)

1.1 Motivation

When we toss a coin, we expect it to land *heads* up with probability 0.5 and *tails* with probability 0.5. According to Diaconis et al. [5, 8, 17], however, a coin toss that starts with heads up results in *heads* with probability more than 0.51, even if the coin is perfect. The results of Diaconis et al. remind us that physical sources of randomness are often biased, and the biases can be unknown. But many applications require random variables with particular probability distributions.

We consider the general problem of simulating a target discrete probability distribution by using a biased source of randomness. For example, we wish to simulate the effect of a fair coin (the target) by tossing a biased coin (the source). More precisely, we say a coin has *bias* p if a coin flip produces *heads* with probability p and *tails* with probability $q = 1 - p$. A coin is *fair* if its bias is $1/2$. Write H and T for the outcomes *heads* and *tails* of a source coin flip, and 0 and 1 for the outcomes of the simulated target coin.

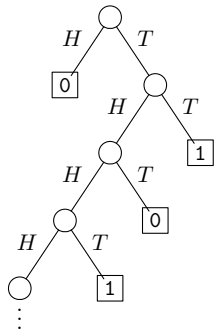


Figure 1.1 Simulating an unbiased coin using a coin of bias 1/3.

Von Neumann [35] presented a simple method for simulating a fair coin using a biased coin: flip the biased coin twice; if the result is HT (respectively TH), then output 0 (1), otherwise repeat the process. Suppose that the bias of the source coin is p. Von Neumann's method simulates a fair coin because $\Pr[HT] = \Pr[TH] = pq$.

An important observation is that von Neumann's method works regardless of the bias p of the source coin because $\Pr[HT] = \Pr[TH] = pq$. In other words, von Neumann's algorithm addresses the case where the bias of random source is *unknown*.

What if we *know* the bias p? For example, suppose that $p = 1/3$. Figure 1.1 shows a binary tree that describes an algorithm for simulating an unbiased coin using a coin of bias 1/3; starting from the root node, follow an edge according to the result of the coin flip, and if a terminal node is reached, output the label 0 or 1. It takes 15/7 coin flips for this algorithm on average to simulate one fair coin flip. Note that von Neumann's method takes $1/(pq)$ coin flips on average to simulate one fair coin flip when the bais of the source coin is p. In this case where $p = 1/3$, $1/(pq) = 9/2$. So the algorithm in Figure 1.1 takes much less coin flips on average than von Neumann's simulation in this case ($p = 1/3$). However, if the source bias p is not 1/3, then the algorithm in Figure 1.1 does not simulate an unbiased coin.

In 1976 Knuth and Yao addressed the problem of simulating a discrete probability distribution using an *unbiased* coin [20]. Their algorithm is based on binary expansions of the target distribution. Despite its simplicity, their algorithm is optimal in the sense that it uses a minimum average number of coin flips. Knuth and Yao posed a question on the case where the source coin is biased. In this thesis, we address the question in two different cases: first, when the bias of the source is *unknown*, and second, when the bias of the source is *known*. In the next sections, we summarize our results and explain their significance in relation to previous work.

1.2 Summary

In Chapter 2, we study the simulation of a given discrete probability distribution using a discrete source of unknown distribution. We study the problem using the concept of *randomizing functions*. Randomizing functions were defined by Elias [10] to study the generation of the uniform distribution. We generalized the concept to the generation of an arbitrary discrete distribution. We define the *efficiency* of randomizing functions to be the average number of output per input of the corresponding simulation of the randomizing function, and *the average cost* to be the reciprocal of the efficiency. It is known that the average cost is bounded below by $\mathcal{H}(\mathbf{r})/\mathcal{H}(\mathbf{p})$, where the source distribution is $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$ and the target distribution $\mathbf{r} = \langle r_1, \ldots, r_m \rangle$, and $\mathcal{H}$ is the Shannon entropy [31]. We call this bound the *entropy bound*. By characterizing randomizing functions, we provide a means to understand such simulations and their optimality in terms of the average cost. Our characterization of randomizing functions allows us to give a better lower bound for the average cost of randomizing functions than the entropy bound. We show that this lower bound is actually tight by constructing randomizing functions that meet the lower bound. So we obtain optimal randomizing functions. In Section 2.4, we

show that the optimal randomizing functions can be computed efficiently. In Section 2.5, we discuss randomizing functions on exhaustive prefix-free sets. We show that the key characterization (Theorem 2.3) does not hold for every exhaustive prefix-free set, and we give a condition on the prefix-free sets for which the optimal randomizing function discussed in the previous sections can be generalized.

In Chapter 3, we study random number generation using a source of known distribution. Our main purpose is to establish a difficulty of finding an optimal simulation. Since we want to show the hardness of the problem, we focus on the simplest case: simulating a coin of bias r using a coin of bias p. Inspired by the work of Knuth and Yao [20], some researchers addressed this problem [1, 13, 29]. However, the optimality question remained open. Knuth and Yao represented their algorithms by binary trees called *discrete distribution generation trees (DDG-trees).* We generalize the DDG-tree so that we can represent algorithms for biased sources. We call a DDG-tree is optimal if its average depth is the smallest among DDG-trees for the given source bias and target bias. We define a model of recursive construction of DDG-trees based on algebraic computations. This model of computation captures all previously known algorithms [1, 13, 20] for this problem. For this model of computation, we use a topological argument to prove that constructing an optimal DDG-tree for a source of known bias is impossible. We also introduce a new method of recursive construction of DDG-trees based on the algebraic computations, which generalizes the Knuth-Yao method. In Section 3.3 we prove that the new method produces optimal DDG-trees in special cases.

DDG-trees can be generalized to simulate the target distribution $\mathbf{r} = \langle r_1, \ldots, r_m \rangle$ using the source distribution is $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$. In Chapter 4, we give an elementary proof of the entropy bound $\mathcal{H}(\mathbf{r})/\mathcal{H}(\mathbf{p})$ for the average depth of a DDG-tree.

The results in Chapter 2 and Chapter 3 were presented at the ACM-SIAM Symposium on Discrete Algorithms in January 2005 [27].

1.3 Related Work

Many previous papers addressed the problem of simulating a discrete probability distribution using another source of randomness. Von Neumann's method may be the earliest known solution for this problem. Hoeffding and Simons [14] and Stout and Warren [33] presented algorithms whose efficiencies are better than von Neumann's [35] for simulating a fair coin from a biased coin.

Elias [10] gave a formal definition of a randomizing procedure for a uniform target distribution, and he designed an infinite sequence of (increasingly complicated) functions whose efficiencies approach arbitrarily close to the entropy bound. Our work in Chapter 2 follows Elias's work, and in fact, the optimal randomizing functions in Section 2.2 are Elias's functions. We prove that Elias's functions are not only nearly optimal in the sense that their efficiencies approach the entropy bound, but also optimal among randomizing functions that accept a fixed length input. We generalize the notion of randomizing function to simulate arbitrary rational distributions, and we address the optimality of such functions. We also demonstrate that Elias's functions can be computed efficiently.

Peres [28] also devised procedures whose efficiencies approach the entropy bound. But unlike Elias's functions, Peres's procedures are not optimal: in particular, his procedure Ψ_3 can be seen as a randomizing function that accepts inputs of length 8, but its efficiency is not optimal. Interestingly, Peres's procedures are defined recursively, and thus they are easily implementable. Although he did not analyze the running time of his procedures, it can be shown that his procedures run in $O(n \log n)$ time with respect to the input size n. This fact inspired us to consider the efficient computability of Elias's functions. Elias did not specify how to compute his functions, and his functions become increasingly complicated as they take longer inputs.

To analyze his procedures, Peres defined a notion of *extracting functions*. We show that extracting functions are special cases of randomizing functions.

Recently, Juels et al. [15] presented optimal algorithms for simulating a fair coin using a source of unknown distribution. Although they did not explicitly say so, their algorithms implement Elias's functions. They proved that their functions are optimal among extracting functions. In contrast, we prove the stronger result that Elias's functions are optimal among randomizing functions. Our proof method differs from the method of Jules et al. Furthermore, unlike Jules et al., we generalize the results to arbitrary target distributions and to functions on prefix-free sets.

Dijkstra [9] presented an elegant algorithm that simulates an m-faced (uniform) roulette from m flips of biased coins, where m is a prime number. His algorithm takes advantage of the fact that m divides $2^m - 2$ when m is a prime number (little Fermat's theorem: $x^m = x \bmod m$), and it generalizes von Neumann's method; note that 2 is a prime! Of course, his method is an example of the randomizing functions that we consider.

Blum [4] considered the problem of simulating an unbiased coin where the source of randomness is a Markov chain whose structure is known, but whose transition probabilities are unknown. This case can be regarded as many sources with unknown distributions (coins with unknown biases in Blum's case), in which the next source to sample is determined by the output value of the current source. Therefore, Blum's problem generalizes our problem for a single source of unknown distribution. If the Markov chain is in the stationary state, then we can regard it as a single source whose outputs are independent of each other. Therefore, randomizing functions can covert the Markov chain source into a particular distribution. Elias [10] discusses the Markov chain source and assumes that the Markov chain is in the stationary state. Blum's algorithm handles a Markov chain source that is not necessarily in the stationary state.

Although von Neumann's algorithm is more than fifty years old, it is still used in a modern computing device, an Intel chipset for random number generation [16]. In that Intel chipset, the bits sampled from an electric source are not independent of each other. Nevertheless, von Neumann's algorithm is used to *correct* the bias.

We will review the details of the method of Knuth and Yao [20] and the method of Han and Hoshi [13] in Section 3.1.3.

Note that the problem we discuss is only remotely related to pseudorandom number generation. Classical pseudorandom generation [19] is about producing a sequence of bits using a deterministic algorithm using very small amount of true randomness, usually called *random seed*, exactly as in the quote in the beginning of this chapter. So the entropy of the resulting sequence is exactly the entropy of the seed, however long the sequence is.

In the modern theory of pseudorandom generation, the focus is on the power of randomness as a resource of computation [12, 24]. Roughly speaking, a bit sequence is *pseudorandom* if efficient algorithms cannot distinguish from true random bits. Therefore, a pseudorandom sequence does not have to be pure independent coin flips. On the other hand, our problem is about generation of *pure* random numbers of a given distribution.

CHAPTER 2

Simulation of Discrete Probability Distribution Using a Source of Unknown Distribution

2.1 Motivation

As introduced in Chapter 1, a coin has *bias* p if a coin flip turns *heads* with probability p and *tails* with probability $q = 1 - p$. A coin is *unbiased* if its bias is $1/2$. We write H and T for the outcomes *heads* and *tails* of a source coin flip, and 0 and 1 for the outcomes of the simulated target coin. We assume that the bias of source coin is p.

Let us consider the function $f_{\mathrm{vN}} : \{H, T\}^2 \to \{0, 1\}$ defined by $f_{\mathrm{vN}}(HT) = 0, f_{\mathrm{vN}}(TH) = 1, f_{\mathrm{vN}}(HH) = f_{\mathrm{vN}}(TT) = \lambda$, where λ is an empty string. Von Neumann's method is abstracted by the function f_{vN}; applying f_{vN} repeatedly is equivalent to von Neumann's method. Again, note that $\Pr[f_{\mathrm{vN}}(x) = 0] = \Pr[f_{\mathrm{vN}}(x) = 1]$ regardless of the source bias p. We will call such a function a *randomizing function*. Intuitively, a randomizing function simulates a discrete probability distribution from a source of *unknown* distribution, and it is immune to the bias of the source as long as the bias is fixed.

As another example, let us consider the function defined in Table 2.1. The function f takes input from $\{H, T\}^3$ (three coin flips) and simulates a binary distribution $\langle 1/3, 2/3 \rangle$. The target distribution does not depend on the source bias because $\Pr[f(x) = 0] =$

Input x	Output $f(x)$
HHH	λ
HHT	0
HTH	1
THH	1
HTT	0
THT	1
TTH	1
TTT	λ

Table 2.1 A Randomizing function for $\langle 1/3, 2/3 \rangle$

$p^2q + pq^2$ and $\Pr[f(x) = 1] = 2(p^2q + pq^2)$, and thus $2 \cdot \Pr[f(x) = 0] = \Pr[f(x) = 1]$ regardless of the source bias p.

We define the *efficiency* of randomizing functions to be the average number of output per input of the corresponding simulation of the randomizing function, and *the average cost* to be the reciprocal of the efficiency. The efficiency of the function f_{vN} is pq, and its average cost is $1/(pq)$. We can show that von Neumann's method is the best—in fact the only way—to simulate a fair coin using *two* flips of a biased coin. When more coin flips are used, the efficiency can be improved. Consider the function shown in Table 2.2; it is a restriction of Peres's function Ψ_2 on $\{H, T\}^4$ [28]. The output of Ψ_2 can be regarded as a fair coin flip, if the output length is 1, or as two independent fair coin flips, if the output length is 2. The average cost of Ψ_2 is

$$\frac{4}{2 \cdot 4p^3q + 1 \cdot 2p^2q^2 + 2 \cdot 4p^2q^2 + 2 \cdot 4pq^3}.$$

That is, Ψ_2 requires approximately two fewer flips of biased coin on average than von Neumann's method, for each p. (See Figure 2.1 on page 18.) Later we will show that this is the best efficiency we can get using four coin flips.

Notice that, for some inputs x, the output length $|\Psi_2(x)|$ is greater than 1. In other words, Ψ_2 maps $\{H, T\}^4$ to $\{0, 1\}^*$. So in our study of randomizing functions, we will

Input x	Output $\Psi_2(x)$
HHHH	λ
HHHT	11
HHTH	01
HTHH	10
THHH	00
HHTT	1
HTHT	11
HTTH	10
THHT	01
THTH	00
TTHH	0
HTTT	10
THTT	00
TTHT	11
TTTH	01
TTTT	λ

Table 2.2 Peres's function Ψ_2

consider functions of the form $f : \{H,T\}^n \to \{0,1\}^*$. In general, we will consider functions of the form $f : \{0,1,\ldots,s-1\}^n \to \{0,1,\ldots,t-1\}^*$ for an s-ary source distribution and a t-ary target distribution.

We will give a formal definition of randomizing functions and study the properties of them. For example, Theorem 2.4 gives a relationship between the output length $|f(x)|$ and the input length $|x|$ of a randomizing function f.

2.2 Randomizing Functions and Optimality: Binary Case

2.2.1 Randomizing Functions

As introduced in Chapter 1, H and T denote *heads* and *tails* from a coin flip of unknown bias; $p = \Pr[H]$, and $q = 1 - p$. When we consider a function $f : \{H, T\}^n \to \{0, 1\}^*$, x denotes an input and z denotes an output. An output $z = f(x)$ may be an empty string λ. Assume that n is fixed throughout this section. For a string z in $\{0, 1\}^*$, let $z[l]$ denote the lth bit of z if $l \leq |z|$, or an empty string λ otherwise; let $z[1, l]$ denote the length-l prefix of z.

We want the output of a function $f : \{H, T\}^n \to \{0, 1\}^*$ to be regarded as a sequence of independent fair coin flips. So f must satisfy (2.1) in the following definition.

Definition 2.1 (Elias [10])**.** *A function $f : \{H, T\}^n \to \{0, 1\}^*$ is* randomizing *if for each l and for each w in $\{0, 1\}^{l-1}$,*

$$\Pr[z[l] = 0 \mid z[1, l-1] = w] = \Pr[z[l] = 1 \mid z[1, l-1] = w] \tag{2.1}$$

for each bias p.

Lemma 2.2. *A function f is randomizing if and only if $\Pr[z[1, l] = w0] = \Pr[z[1, l] = w1]$, for every w in $\{0, 1\}^*$.*

Proof. Clearly,

$$\begin{aligned} &\Pr[z[l] = i \mid z[1, l-1] = w] \\ &= \Pr[z[1, l-1] = w \text{ and } z[l] = i] / \Pr[z[1, l-1] = w] \\ &= \Pr[z[1, l] = wi] / \Pr[z[1, l-1] = w] \end{aligned}$$

for $i = 0, 1$. The statement of the lemma follows. □

Let B_w be the set of x such that w is a prefix of $f(x)$, and let C_w be $f^{-1}(w)$:

$$\begin{aligned} B_w &= \{x \in \{H,T\}^n \mid f(x)[1,|w|] = w\} \\ C_w &= \{x \in \{H,T\}^n \mid f(x) = w\} \end{aligned}$$

Then we have a disjoint union

$$\{H,T\}^n = B_0 \cup B_1 \cup C_\lambda, \tag{2.2}$$

and for each w in $\{0,1\}^*$, we have a disjoint union

$$B_w = B_{w0} \cup B_{w1} \cup C_w. \tag{2.3}$$

By convention, let us write $B_\lambda = \{H,T\}^n$. If we restrict f to be randomizing, then we can say much more than (2.3). According to Lemma 2.2, f is randomizing if and only if for each $w \in \{0,1\}^*$,

$$\Pr[x \in B_{w0}] = \Pr[x \in B_{w1}]. \tag{2.4}$$

Let $S_{(n,k)}$ be the set of all strings x in $\{H,T\}^n$ such that the number of T's in x is k. We write S_k for $S_{(n,k)}$, when the context is clear.

Theorem 2.3. *If a function f is randomizing, then for each w in $\{0,1\}^*$, $|B_{w0}| = |B_{w1}|$. Moreover, for each k, $|B_{w0} \cap S_k| = |B_{w1} \cap S_k|$.*

Proof. By Lemma 2.2, $\Pr[x \in B_{w0}] = \Pr[x \in B_{w1}]$, for each w in $\{0,1\}^*$. Note that, for u in $\{0,1\}^*$, $\Pr[x \in B_u] = \sum_{i=0}^n |B_u \cap S_i| p^{n-i} q^i$. So the equality is written as

$$\sum_{i=0}^{n} \left(|B_{w0} \cap S_i| - |B_{w1} \cap S_i|\right) p^{n-i} q^i = 0. \tag{2.5}$$

We can show that the set of polynomials $\{y^n, y^{n-1}(1-y), \ldots, (1-y)^n\}$ is linearly independent in the vector space of functions on $[0,1]$. (See Proposition 2.19.) Since (2.5) holds for each p in $[0,1]$ and $q = 1-p$, we conclude that $|B_{w0} \cap S_i| = |B_{w1} \cap S_i|$, for each i, and thus $|B_{w0}| = |B_{w1}|$. □

In the proof of Theorem 2.3, we used the fact that a randomizing function needs to satisfy condition (2.1) for each bias p; if there is k such that $|B_{w0} \cap S_k| \neq |B_{w1} \cap S_k|$, then there are only finitely many values of p that satisfy equation (2.5).

In Table 2.2, $\{H,T\}^4$ is partitioned into subsets $S_{(4,k)}$, where $k = 0, 1, 2, 3, 4$. We can see that Theorem 2.3 holds for the function defined by the table.

As an application of the characterization of randomizing function given in Theorem 2.3, we can prove a relationship between the output length and the input length for randomizing functions.

Theorem 2.4. *Let $f : \{H,T\}^n \to \{0,1\}^*$ be n-randomizing. If $x \in S_k$, then $|f(x)| \leq \lfloor \log_2 \binom{n}{k} \rfloor$.*

Proof. Fix k. Let $B'_w = B_w \cap S_k$ and $C'_w = C_w \cap S_k$. From (2.2) and (2.3),

$$S_k = B'_0 \cup B'_1 \cup C'_\lambda, \tag{2.6}$$

and

$$B'_w = B'_{w0} \cup B'_{w1} \cup C'_w, \tag{2.7}$$

for each $w \in \{0,1\}^*$. Suppose that B'_{u0} is nonempty. By Theorem 2.3 we have $|B'_{u0}| = |B'_{u1}|$. So by (2.7), we conclude that

$$|B'_u| \geq 2|B'_{u0}| = 2|B'_{u1}|.$$

Again, since B'_u is not empty, using induction, we conclude that $\binom{n}{k} = |S_k| = |B'_\lambda| \geq 2^{|u0|}|B_{u0}| = 2^{|u1|}|B_{u1}|$, and thus

$$2^{|u0|} = 2^{|u1|} \leq \binom{n}{k}/|B_{u0}| = \binom{n}{k}/|B_{u1}|. \tag{2.8}$$

Let v be an output of f on S_k of maximum length. Then (2.8) is rewritten for v as

$$2^{|v|} \leq \binom{n}{k}/|B_v|.$$

Since B_v is nonempty, $|B_v| \geq 1$. Therefore, if $x \in S_k$, then $|f(x)| \leq |v| \leq |\log_2 \binom{n}{k}|$. □

Corollary 2.5. *Let f be an n-randomizing function, then for every $x \in \{H,T\}^n$,*

$$|f(x)| \le \lfloor \log_2 \binom{n}{\lfloor n/2 \rfloor} \rfloor.$$

This corollary gives an upper bound of effciency of randomizing functions.

2.2.2 Extracting Functions

Before we discuss the optimality of randomizing functions, let us introduce extracting functions as a special case. Extracting functions are a natural subclass of randomizing functions. In fact, most functions considered in the literature (including [10, 15, 28]) are extracting functions. Not every randomizing function is extracting, however; see the example after Lemma 2.7.

Definition 2.6 (Peres [28])**.** *A function $f : \{H,T\}^n \to \{0,1\}^*$ is* extracting *if for each bias p and each pair z_1, z_2 in $\{0,1\}^*$ such that $|z_1| = |z_2|$, $\Pr[f(x) = z_1] = \Pr[f(x) = z_2]$. In other words, f is uniformly distributed in $\{0,1\}^l$ whenever f has a value in $\{0,1\}^l$, for each l and for each bias p.*

Von Neumann's function and Peres's function Ψ_2 are extracting functions.

Lemma 2.7. *Every extracting function is randomizing.*

Proof. If f is extracting, then $\Pr[z = w0u] = \Pr[z = w1u']$ for every w, u, u' in $\{0,1\}^*$ such that $|u| = |u'|$. Therefore an extracting function satisfies the condition

$$\Pr[z[1,l] = w0] = \Pr[z[1,l] = w1]. \tag{2.9}$$

So we have

$$\begin{aligned} \Pr[z[l] = 0 \,|\, z[1,l-1] = w] &= \Pr[z[1,l] = w0] / \Pr[z[1,l-1] = w] \\ &= \Pr[z[1,l] = w1] / \Pr[z[1,l-1] = w] \\ &= \Pr[z[l] = 1 \,|\, z[1,l-1] = w]. \end{aligned}$$

Therefore an extracting function is randomizing. □

Example The converse of Lemma 2.7 is not true. Consider the function f on $\{H,T\}^4$ such that

$$\begin{aligned} S_{(4,1)} &\mapsto 00\ 01\ 10\ 11 \\ S_{(4,2)} &\mapsto 00\ 01\ 10\ 11\ 0\ 1 \\ S_{(4,3)} &\mapsto 10\ 11\ 0\ 0. \end{aligned}$$

This function f is not extracting since its restriction on $S_{(4,3)}$ is not uniformly distributed in $\{0,1\}^1$ and $\{0,1\}^2$. However, f is randomizing.

A function $f : \{H,T\}^n \to \{0,1\}^*$ is *perfectly extracting* if $f|_{S_k}$, the restriction of f to S_k, is uniformly distributed in $\{0,1\}^l$ whenever $f|_{S_k}$ has a value in $\{0,1\}^l$, for each l. In Table 2.2, Peres's function Ψ_2 is perfectly extracting.

Lemma 2.8. *A function $f : \{H,T\}^n \to \{0,1\}^*$ is extracting if and only if f is perfectly extracting.*

Proof. It is clear that every perfectly extracting function is extracting. We will prove the other direction. If f is extracting, then $\Pr[f(x) = z_1] = \Pr[f(x) = z_2]$ for all p. Equivalently,

$$\sum_{i=0}^{n} \left(|f^{-1}(z_1) \cap S_i| - |f^{-1}(z_2) \cap S_i| \right) p^{n-i} q^i = 0. \tag{2.10}$$

As in the proof of Theorem 2.3, using the linear independence of $\{y^n, y^{n-1}(1-y), \ldots, (1-y)^n\}$ (Proposition 2.19), we conclude that $|f^{-1}(z_1) \cap S_k| = |f^{-1}(z_2) \cap S_k|$ for each k. □

2.2.3 Optimal Randomizing Functions

Using the characterization of randomizing functions in Theorem 2.3, we can now identify the optimal randomizing functions.

Elias [10] defined the function E^n on $\{H,T\}^n$ as follows. Let $a_d 2^d + a_{d-1}2^{d-1} + \cdots + a_0$ be the binary expansion of $|S_k|$. Partition S_k into subsets S_{ki} of size 2^i, for i such that $a_i = 1$. If $a_i = 0$, then $S_{ki} = \emptyset$. The function E^n maps each string in S_{ki} to a unique string in $\{0,1\}^i$. So $E^n(S_{ki}) = \{0,1\}^i$, and $E^n(S_k) = \bigcup_{i,a_i=1}\{0,1\}^i$. If $a_0 = 1$, that is, if $|S_k|$ is odd, then there is one string in S_k for which E^n outputs a null string. Note that Elias did not specify how to partition S_k nor how to map a string in S_{ki} to a string in $\{0,1\}^i$. Peres's function shown in Table 2.2 can serve as E^4.

Restricted to S_k, according to Theorem 2.3, a randomizing function generates $w0$ and $w1$ on the same number of inputs. This strong constraint on randomizing functions enables us to prove that E^n is optimal. Define $\alpha(N) = \sum n_k 2^{n_k}$, where $\sum 2^{n_k}$ is the standard binary expansion of N. Since E^n outputs i bits for each string in S_{ki}, and there are 2^i strings in S_{ki}, the sum of output lengths over S_{ki} is $\alpha(|S_{ki}|)$. The average output length of E^n is therefore $\frac{1}{n}\sum_{k=0}^{n}\alpha(|S_{ki}|)p^{n-k}q^k$. The function α is monotone increasing, and $2\alpha(N) \le \alpha(2N)$; in fact, $\alpha(2N) - 2\alpha(N) = 2N$. We prove a more general statement, Lemma 2.15 below.

Lemma 2.9. *Let $g : S \to \{0,1\}^*$ satisfy the condition $|D_{w0}| = |D_{w1}|$, for each $w \in \{0,1\}^*$, where $D_u = \{x \in S \mid g(x)[1,|u|] = u\}$. Then*

$$\sum_{x\in S} |g(x)| \le \alpha(|S|). \tag{2.11}$$

Proof. We proceed by induction on $|S|$. Note that $|D_0| = |D_1| \le |S|/2$. When $|S| = 1$, $|D_0| = |D_1| = 0$, and $\alpha(1) = 0$. So the inequality (2.11) holds.

When $|S| > 1$, since $|D_0| = |D_1| < |S|$. Note that $g|_{D_0}$ and $g|_{D_1}$ satisfy the condition of the lemma. Hence, by induction hypothesis,

$$\sum_{x\in D_0} |g(x)| = \sum_{x\in D_1} |g(x)| \le \alpha(|D_0|).$$

Thus, $\sum_{x\in S}|g(x)| = \sum_{x\in D_0}|g(x)| + \sum_{x\in D_1}|g(x)| \le 2\alpha(|D_0|) \le \alpha(2|D_0|) \le \alpha(|S|)$. □

Theorem 2.10. *The function E^n is optimal among all randomizing functions on $\{H,T\}^n$. Also E^n is optimal among all extracting functions on $\{H,T\}^n$.*

Proof. By definition, E^n is perfectly extracting. Therefore, E^n is extracting, and thus, randomizing.

By Theorem 2.3 and Lemma 2.9, if f is randomizing, then $\sum_{x\in S_k}|f(x)| \le \alpha(|S_k|)$. The average output length of f is, therefore,

$$\frac{1}{n}\sum_{k=0}^{n}\sum_{x\in S_k}|f(x)|p^{n-k}q^k \le \frac{1}{n}\sum_{k=0}^{n}\alpha(|S_k|)p^{n-k}q^k.$$

So we have completed the proof of the theorem. □

By Theorem 2.10, von Neumann's function is 2-optimal, and Ψ_2 is 4-optimal.

Figure 2.1 shows the average costs of optimal n-randomizing functions for a few values of n and the entropy bound.

2.3 From s-Faced Dice to t-Faced Dice

As we noted earlier, Dijkstra [9] presented an elegant generalization of von Neumann's method that simulates an m-faced (uniform) die from m flips of biased coins, where m is a prime number. Although Dijkstra's method is elegant and easy to compute, its usage of the random source is rather inefficient. It takes more than m coin flips to obtain one output, or about $\log_2 m$ bits, on average. Fortunately, our discussion in Section 2.2 generalizes to this case, and the function E^n generalizes to simulate a uniform distribution over $\{0,1,\ldots,m-1\}$, where m is any positive integer. It will be clear that the generalized function is optimal and efficiently computable.

In fact, the generalization does not stop there. The characterization of randomizing functions generalizes to the problem of simulating an *arbitrary rational distribution*, not

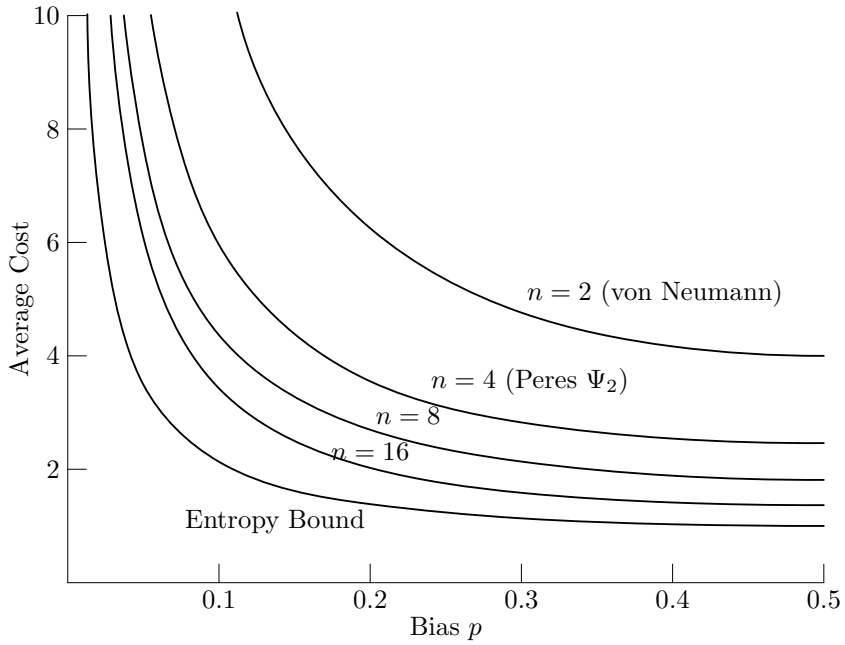


Figure 2.1 Average Cost of Optimal Randomizing Functions and Entropy Bound

necessarily uniform, using a *many-faced die* of unknown distribution as a random source. Note that our generalization is twofold: With respect to the target, we generalize from a fair coin to an arbitrary rational distribution. With respect to the source, we generalized from a coin to a many-faced die. The optimal randomizing function for a binary source and a binary target that we discussed in Section 2.2 will generalize so that we obtain an optimal function that simulates an arbitrary rational distribution from a many-faced die. We will show that no randomizing function can simulate an irrational distribution (such that the probability of an output symbol is an irrational number). So our result is the best that we can obtain to simulate a discrete distribution from a source of unknown distribution.

Let us say that we want to simulate a t-faced die using an s-faced die. Let $\langle r_0, \ldots, r_{t-1} \rangle$ be the target distribution, where each r_i is a rational number, and let $\langle p_0, \ldots, p_{s-1} \rangle$ be the unknown source distribution. We will call a function from $\{0, 1, \ldots, s-1\}^n$ to

$\{0, 1, \ldots, t-1\}^*$ whose output simulates the target distribution a *randomizing function* for $\langle r_0, \ldots, r_{t-1} \rangle$. The optimal function for the simulation that we will describe is a generalization of E^n. First, let m be a fixed integer no less than 2. Then we construct a function $E_s^n : \{0, 1, \ldots, s-1\}^n \to \{0, 1, \ldots, m-1\}^*$, such that E_s^n simulates a uniform m-faced die. The function E_s^n is optimal. Then, for positive integers $m_0, m_1, \ldots, m_{t-1}$ such that $m_0 + m_1 + \cdots + m_{t-1} = m$, consider

$$h(x) = \begin{cases} 0 & \text{if } 0 \leq x < m_0 \\ 1 & \text{if } m_0 \leq x < m_0 + m_1 \\ \cdots & \\ t-1 & \text{if } m_0 + \cdots + m_{t-2} \leq x < m \end{cases}$$

Then consider $\tilde{E}_s^n = \tilde{h} \circ E_s^n$, where $\tilde{h} : \{0, \ldots, m-1\}^* \to \{0, \ldots, t-1\}^*$ is defined $\tilde{h}(u_1 \cdots u_k) = h(u_1) \cdots h(u_k)$. The function $\tilde{E}_s^U$, then, simulates a rational distribution $\langle m_0/m, \ldots, m_{t-1}/m \rangle$. By taking m as a common denominator of $r_0, \ldots, r_{t-1}$, we can choose m_i such that $r_i = m_i/m$ for each i. In Section 2.3.2, we prove that $\tilde{E}_s^n$ is optimal among the randomizing functions for $\langle r_0, \ldots, r_{t-1} \rangle$.

2.3.1 The Function E_s^n

Let $S_{(d_0,\ldots,d_{s-1})}$ be the subset of $\{0, 1, \ldots, s-1\}^n$ such that the number of $0, 1, \ldots, s-1$ in its elements are $d_0, d_1, \ldots, d_{s-1}$, respectively. For the sake of notational convenience, for nonnegative integers $d_0, \ldots, d_{s-1}$, let us write $\mathbf{d} = (d_0, \ldots, d_{s-1})$, $|\mathbf{d}| = d_0 + \cdots + d_{s-1}$, and $S_{\mathbf{d}} = S_{(d_0,\ldots,d_{s-1})}$. Let $a_k m^k + a_{k-1} m^{k-1} + \cdots + a_0$ be the m-ary expansion of $|S_{\mathbf{d}}|$. Partition $S_{\mathbf{d}}$ into subsets $S_{\mathbf{d}i}$ of size $a_i m^i$, where $0 < a_i \leq m$. Then E_s^n outputs a_i copies of $\{0, 1, \ldots, m-1\}^i$ on $S_{\mathbf{d}i}$. We will show that E_s^n is efficiently computable in Section 2.4.

Let us call $f : \{0, 1, \ldots, s-1\}^n \to \{0, 1, \ldots, m-1\}^*$ a *uniformly randomizing function* over $\{0, 1, \ldots, m-1\}$ if

$$\Pr[z[l] = i \mid z[1, l-1] = w] = \Pr[z[l] = j \mid z[1, l-1] = w] \tag{2.12}$$

for each l and each w in $\{0, 1, \ldots, m-1\}^{l-1}$. Define $\alpha_m(N) = \sum n_i a_i m^{n_i}$, where $\sum a_i m^{n_i}$ is the m-ary expansion of N. (Note that the function α in Section 2.2 is α_2.) Then the average number of output digits of E_s^n is

$$\frac{1}{n} \sum_{|\mathbf{d}|=n} \alpha_m(|S_{\mathbf{d}}|) p_0^{d_0} \cdots p_{s-1}^{d_{s-1}}.$$

In Section 2.3.2, we generalize Theorem 2.3 and Lemma 2.9 to $\{0, 1, \ldots, m-1\}$, and we show that E_s^n is optimal among uniformly randomizing functions over $\{0, 1, \ldots, m-1\}$.

2.3.2 Randomizing Functions and Optimality of $\tilde{E}_s^n$

Now let us discuss the generation of an arbitrary rational distribution. We generalize Definition 2.1 as follows.

Definition 2.11. *A function $f : \{0, \ldots, s-1\}^n \to \{0, \ldots, t-1\}^*$ is* randomizing *for a distribution $\langle r_0, \ldots, r_{t-1} \rangle$ if for each l and for each w in $\{0, \ldots, t-1\}^{l-1}$,*

$$\Pr[z[l] = i | z[1, l-1] = w] = r_i. \tag{2.13}$$

Lemma 2.12. *A function $f : \{0, \ldots, s-1\}^n \to \{0, \ldots, t-1\}^*$ is randomizing for a distribution $\langle r_0, \ldots, r_{t-1} \rangle$ if and only if*

$$\frac{\Pr[z[1, l] = wi]}{\Pr[z[1, l] = wj]} = \frac{r_i}{r_j}. \tag{2.14}$$

Let $B_w = \{x \in \{0, \ldots, s-1\}^n \mid f(x)[1, |w|] = w\}$ as earlier. Then we have

$$\{0, \ldots, s-1\}^n = B_0 \cup B_1 \cup \cdots \cup B_{t-1} \cup C_\lambda,$$

and

$$B_w = B_{w0} \cup B_{w1} \cup \cdots \cup B_{w(t-1)} \cup C_w.$$

Like Theorem 2.3, the following theorem characterizes randomizing functions for rational distributions: Restricted to $S_{\mathbf{d}}$, a randomizing function generates wi, for each w and $i = 0, \ldots, t-1$, with respect to the proportion $\langle r_0, \ldots, r_{t-1} \rangle$.

Theorem 2.13. *If f is randomizing, then for each i and j, and for each w,*

$$\frac{|B_{wi} \cap S_{\mathbf{d}}|}{|B_{wj} \cap S_{\mathbf{d}}|} = \frac{r_i}{r_j}. \tag{2.15}$$

Proof. As in the proof of Theorem 2.3, the condition (2.14) gives the equation

$$\sum_{|\mathbf{d}|=n} \left(r_j |B_{wi} \cap S_{\mathbf{d}}| - r_i |B_{wj} \cap S_{\mathbf{d}}| \right) p_0^{d_0} \cdots p_{s-1}^{d_{s-1}} = 0. \tag{2.16}$$

The equation holds regardless of $p_0, \ldots, p_{s-1}$ such that $p_0 + \cdots + p_{s-1} = 1$. The set of polynomials that we obtain by substituting y_i for p_i, $0 \le i \le s-2$, and by substituting $1 - y_0 - y_1 - \cdots - y_{s-2}$ for p_{s-1}, is linearly independent (Proposition 2.20). Therefore the coefficients $r_j |B_{wi} \cap S_{\mathbf{d}}| - r_i |B_{wj} \cap S_{\mathbf{d}}|$ in (2.16) are zero. □

Note that, for each i, the set $B_{wi} \cap S_{\mathbf{d}}$ is finite. Therefore, as a corollary, we obtain the following.

Corollary 2.14. *There is no randomizing function for an irrational target distribution.*

In order to prove the optimality of $\tilde{E}_s^n$, we need to use some properties of α_m. The function α_m is clearly increasing.

Lemma 2.15. *For positive integers N_1 and N_2, we have*

$$\alpha_m(N_1) + \alpha_m(N_2) \le \alpha_m(N_1 + N_2). \tag{2.17}$$

Proof. It is easy to show that $\alpha_m(\sum_i d_i m^i) = \sum_i \alpha_m(d_i m^i)$, if $0 \leq d_i < m$ for each i. If $a + b \geq m$ so that $a + b = m + c$, where a, b and c are integers such that $0 \leq a, b, c < m$, then

$$\begin{aligned} \alpha_m(am^i + bm^i) &= \alpha_m(m^{i+1} + cm^i) \\ &= (i+1)m^{i+1} + icm^i \\ &= m^{i+1} + \alpha_m(am^i) + \alpha_m(bm^i). \end{aligned}$$

So if $a + b \geq m$, then $\alpha_m(am^i + bm^i) - \alpha_m(am^i) - \alpha_m(bm^i) = m^{i+1}$. Let $N_1 = \sum_{i=0}^{k} a_i m^i$ and $N_2 = \sum_{i=0}^{l} b_i m^i$ be m-ary expansions. Therefore, $\alpha_m(N_1 + N_2) - \alpha_m(N_1) - \alpha_m(N_2) = \sum_{i \in C} m^{i+1}$, where $C = \{i \mid a_i + b_i \geq m\}$, and the lemma holds. □

Lemma 2.16. *Let $g : S \to \{0, 1, \ldots, t-1\}^*$ satisfy the condition $|D_{wi}|/|D_{wj}| = m_i/m_j$, for each $w \in \{0, 1, \ldots, t-1\}^*$, where $D_u = \{x \in S \mid g(x)[1, |u|] = u\}$, $i = 0, 1, \ldots, t-1$. Then*

$$\sum_{x \in S} |g(x)| \leq \alpha_m(|S|). \tag{2.18}$$

Proof. We proceed by induction on $|S|$. When $|S| = 1$, $|D_i| = 0$, for each i. So the inequality (2.18) holds.

When $|S| > 1$, we have $|D_0| + \cdots + |D_{t-1}| \leq |S|$. Note that each $|D_i|$ is positive, since each m_i is positive. So $|D_i| < |S|$, for each i. Since $g|_{D_i}$ satisfies the condition of the lemma, by the induction hypothesis,

$$\sum_{x \in D_i} |g(x)| \leq \alpha_m(|D_i|),$$

for each i. Thus

$$\sum_{x \in S} |g(x)| = \sum_{x \in D_0} |g(x)| + \cdots + \sum_{x \in D_{t-1}} |g(x)|$$

$$
\begin{aligned}
&\leq \alpha_m(|D_0|) + \cdots + \alpha_m(|D_{t-1}|) \\
&\leq \alpha_m(|D_0| + \cdots + |D_{t-1}|) \\
&\leq \alpha_m(|S|).
\end{aligned}
$$

□

So, if f is randomizing for $\langle m_0/m, \ldots, m_{t-1}/m \rangle$, then

$$\sum_{x \in S_{\mathbf{d}}} |f(x)| \leq \alpha_m(|S_{\mathbf{d}}|).$$

The average output length of f is, therefore,

$$
\begin{aligned}
&\frac{1}{n} \sum_{|\mathbf{d}|=n} \sum_{x \in S_{\mathbf{d}}} |f(x)| p_0^{d_0} \cdots p_{s-1}^{d_{s-1}} \\
&\quad \leq \frac{1}{n} \sum_{|\mathbf{d}|=n} \alpha_m(|S_{\mathbf{d}}|) p_0^{d_0} \cdots p_{s-1}^{d_{s-1}},
\end{aligned}
$$

which is the average output length of $\tilde{E}_s^n$. We conclude that $\tilde{E}_s^n$ has the largest average output length among randomizing functions for this target distribution.

Theorem 2.17. *The function $\tilde{E}_s^n$ is optimal among randomizing functions for $\langle m_0/m, \ldots, m_{t-1}/m \rangle$.*

The function f defined by Table 2.1 is an optimal randomizing function for the target distribution $\langle 1/3, 2/3 \rangle$ for which $n = 3, s = 2, t = 2$.

2.4 Computation of Optimal Randomizing Functions

As we have seen in the previous sections, a randomizing function is characterized by its behavior on the sets $S_{(n,k)}$ or $S_{\mathbf{d}}$. Naturally, these sets play a critical role in the computation of optimal randomizing functions. We first discuss the computation of optimal randomizing functions on the sets of fixed-length inputs; we show that E^n and E_s^n are computable in polynomial time in n.

2.4.1 Computation of E^n

Theorem 2.18. *Elias's function E^n is computable in polynomial time.*

Proof. The idea is the following:

1. Given an input x in $\{H,T\}^n$, count the number of T's in x. Let k be the number of T's. So x is in $S_{(n,k)}$.
2. Compute the rank $r(x)$ of x in $S_{(n,k)}$, with respect to the lexicographical order.
3. Using the rank $r(x)$, determine the output string: Suppose that $|S_{(n,k)}| = 2^{n_1} + 2^{n_2} + \cdots + 2^{n_j} + a_0$, where $n_1 > n_2 > \cdots > n_j$ and $a_0 = 1$ if $|S_{(n,k)}|$ is odd, $a_0 = 0$ otherwise. If $r(x) \leq 2^{n_1}$, then $f(x)$ is the n_1-digit binary representation of x. If $2^{n_1} + \cdots + 2^{n_i} < r(x) \leq 2^{n_1} + \cdots + 2^{n_{i+1}}$, then $f(x)$ is n_{i+1}-digit binary representation of $r(x) - 2^{n_1} + \cdots + 2^{n_i}$.

The first and third steps can be computed efficiently. The second step, the problem of computing rank in $S_{(n,k)}$, was also considered in a different context of enumerative source coding [6, 30], and in the context of generation of all combinations [18] implicitly. Using the recursive structure of $S_{(n,k)}$, it is not hard to prove the following. Let $c_1, c_2, \ldots, c_k$ be the positions of T's in x, counted from the right and regarding the rightmost digit as 0th, such that $0 \leq c_1 < c_2 < \ldots < c_k < n$. Then

$$r(x) = \binom{c_k}{k} + \binom{c_{k-1}}{k-1} + \cdots + \binom{c_1}{1}. \tag{2.19}$$

Since the binomial coefficients in (2.19) are computable in $O(n\mu(\log_2 n))$ time, where $\mu(t)$ is the time complexity of multiplication of t-digit binary numbers, r is computable in $O(kn\mu(\log_2 n))$ time. □

Input x	Rank $r(x)$
$0\cdots0\overbrace{1\cdots1}^{k}$	0
$0\cdots010\overbrace{1\cdots1}^{k-1}$	1
$0\cdots0110\overbrace{1\cdots1}^{k-2}$	2
$\vdots$	
$0\cdots0\overbrace{1\cdots1}^{k}0$	$\binom{k}{1}$
$0\cdots0100\overbrace{1\cdots1}^{k-1}$	$\binom{k}{1}+1$
$\vdots$	
$0\cdots0\overbrace{1\cdots1}^{k}00$	$\binom{k}{1}+\binom{k+1}{2}$
$\vdots$	
$10\cdots0\overbrace{1\cdots1}^{k-1}$	$\binom{k}{1}+\binom{k+1}{2}+\cdots+\binom{n-2}{n-k-1}+1$
$\vdots$	
$\overbrace{1\cdots1}^{k}0\cdots0$	$\binom{k}{1}+\binom{k+1}{2}+\cdots+\binom{n-2}{n-k-1}+\binom{n-1}{n-k}$

Table 2.3 Rank table of $S_{(n,k)}$

Computation of Ranking We can compute the ranking without using the formula (2.19). Table 2.3 shows the ranks of strings in $S_{(n,k)}$. Note that, in this table and the following discussion, we use 0 and 1, instead of H and T. The table is divided into $n-k+1$ categories. The lth category has $(l-1)$ 0's after the first 1 in the string, and it has one-to-one correspondence to the set $S_{(l+k-2,k-1)}$ of strings with $(l-1)$ 0's and $(k-1)$ 1's. So there are $\binom{k+l-2}{k-1}$ strings in the lth category. Since the sum of the strings in each category is the same as the strings in $S_{(n,k)}$, we have the identity

$$\binom{n}{k} = \binom{k-1}{0}+\binom{k}{1}+\binom{k+1}{2}+\cdots+\binom{n-1}{n-k}$$

$$= \binom{k-1}{k-1} + \binom{k}{k-1} + \binom{k+1}{k-1} + \cdots + \binom{n-1}{k-1}.$$

The structure of the table is recursive; we noted that the lth category has a one-to-one correspondence with $S_{(l+k-2,k-1)}$. Now the lth category is again divided into subcategories in the same way, as $S_{(n,k)}$ was divided, according to the number of 0's after the second 1. The rank function r can be computed in a similar pattern. Suppose that x is a string in lth category. Then the rank $r(x)$ is the rank of the first string in the category plus the rank of corresponding substring in $S_{(l+k-2,k-1)}$.

The following example illustrates the idea of computing the rank function.

$$\begin{aligned} r(010010100) &= r(10010100) \\ &= r(10000011) + r(10100) \\ &= r(10000011) + r(10001) + r(100) \\ &= \binom{7}{3} + \binom{4}{2} + \binom{2}{1} \\ &= 35 + 6 + 2 = 43. \end{aligned}$$

Note that the rank of the first string in the lth category is $\binom{k+l-2}{k}$, the number of combinations with $l-2$ 0's and k 1's. As shown above, the computation of $r(x)$ is reduced to the computation of the rank of the first substring in each subcategory, and the first substring is exactly the substring starting with 1. Since there are k such substrings, and their rank is a binomial coefficient, which is computable in polynomial time, the function r is polynomial-time computable.

Knuth [18] discusses generation of all combinations. In [18], his algorithm (Algorithm T) generates $S_{(n,k)}$ in the lexicographical order and his Theorem L states that x is generated after exactly $r(x)$ other strings are generated. The above discussion on the computation of the rank function $r(x)$ solves an inverse problem, and also gives another proof of (2.19).

2.4.2 Computation of $\tilde{E}_s^n$

Since the computation of the function $\tilde{h}$ is easy, we only need to show how to compute E_s^n. Like E^n, we can compute E_s^n as follows:

1. Given an input x in $\{0,\ldots,s-1\}^n$, count the number d_i of i in x, for each $i=0,\ldots,s-1$. Then x is in $S_{\mathbf{d}}$, where $\mathbf{d}=(d_0,\ldots,d_{s-1})$.
2. Compute $r(x)$, where $r: S_{\mathbf{d}} \to \{0,1,\ldots,|S_{\mathbf{d}}|-1\}$, is a one-to-one function.
3. Using $r(x)$, determine the output string: Suppose that $|S_{\mathbf{d}}| = a_k m^k + a_{k-1}m^{k-1} + \cdots + a_0$ is the m-ary expansion. If $r(x) \le a_k m^k$, then $E_s^n(x)$ is k-digit m-ary representation of $r(x) \mod m^k$. If $a_k m^k + \cdots + a_{i+1}m^{i+1} < r(x) \le a_k m^k + \cdots + a_i m^i$ and $a_i \neq 0$, then $E_s^n(x)$ is i-digit m-ary representation of $r(x) - a_k m^k + \cdots + a_{i+1}m^{i+1} \mod m^i$.

We only need to show that the rank function is efficiently computable. Although it is possible to compute the lexicographic rank (as in [6] and [30]), we will consider a different order; in order to compute E_s^n, we only need a unique value in $\{0,1,\ldots,|S_{\mathbf{d}}|-1\}$ for each string in $S_{\mathbf{d}}$.

We partition $S_{\mathbf{d}}$ recursively. The number of the digit $s-1$ in the strings of $S_{\mathbf{d}}$ is d_{s-1}. Partition $S_{\mathbf{d}}$ into $\binom{n}{d_{s-1}}$ distinct subsets $S(c_1,\ldots,c_{d_{s-1}})$ with respect to the positions of $s-1$: $S(c_1,\ldots,c_{d_{s-1}})$ is the set of strings whose positions of the digit $s-1$ are $c_1, c_2, \ldots, c_{d_{s-1}}$ counted from the right and regarding the rightmost digit as 0th, such that $0 \le c_1 < c_2 < \ldots < c_{d_{s-1}} < n$. Note that for each $(c_1, c_2, \ldots, c_{d_{s-1}})$,

$$|S(c_1, c_2, \ldots, c_{d_{s-1}})| = \binom{n-d_{s-1}}{d_0\, d_1 \cdots d_{s-2}}.$$

Now each $S(c_1, c_2, \ldots, c_{d_{s-1}})$ is isomorphic to the set $S_{(d_0,d_1,\ldots,d_{s-2})} \subset \{0,1,\ldots,s-2\}^{n-d_{s-1}}$. So the same argument applies to $S(c_1, c_2, \ldots, c_{d_{s-1}})$, and we can partition it into $\binom{n-d_{s-1}}{d_{s-2}}$

disjoint subsets according to the positions of the digit $s-2$. By proceeding this way we obtain the well-known formula

$$\binom{n}{d_0\, d_1 \cdots\, d_{s-1}} = \binom{n}{d_{s-1}}\binom{n-d_{s-1}}{d_{s-2}}\cdots\binom{d_0+d_1}{d_0}.$$

Now we consider a natural order that reflects this recursive structure of $S_{\mathbf{d}}$. For a given x in $\{0,1,\ldots,s-1\}^n$, let $\beta_i(x)$ be the string obtained by deleting digits greater than i in x. Then let $b_i(x)$ be the binary string whose positions of 1 are the positions of the digit i in $\beta_i(x)$. For example, $b_3(231130321) = 010010100$, and $b_1(231130321) = 1101$. Let us consider the mapping

$$\rho : S_{\mathbf{d}} \to \left[\binom{n}{d_{s-1}}\right] \times \left[\binom{n-d_{s-1}}{d_{s-2}}\right] \times \cdots \times \left[\binom{d_0+d_1}{d_0}\right]$$

defined by $\rho(x) = (r(b_{s-1}(x)), r(b_{s-2}(x)), \ldots, r(b_1(x)))$, where $[N] = \{0,1,\ldots,N-1\}$, and r is the rank function defined on the binary sequences. Define $x \prec y$ if $\rho(x)$ precedes $\rho(y)$ in the lexicographical order. Note that $\prec$ is not the lexicographical order on $S_{\mathbf{d}}$. Via the natural correspondence (which is order-preserving)

$$\gamma : \left[\binom{n}{d_{s-1}}\right] \times \left[\binom{n-d_{s-1}}{d_{s-2}}\right] \times \cdots \times \left[\binom{d_0+d_1}{d_0}\right] \to \left[\binom{n}{d_0\, d_1 \cdots\, d_{s-1}}\right],$$

we define the rank function $r(x) = \gamma(\rho(x))$. Now keeping the above correspondence in mind, we can compute $r(x)$ recursively as follows:

$$r(x) = r(b_{s-1}(x))\binom{n}{d_0 \cdots\, d_{s-2}} + r(\beta_{s-1}(x)).$$

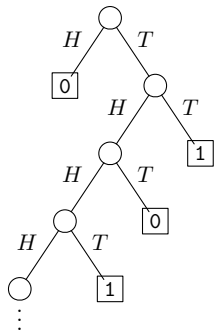
For example,

$$\begin{aligned} r(231130321) &= r(010010100)\binom{6}{1\,3\,2} + r(211021) \\ &= r(010010100)\binom{6}{1\,3\,2} + r(100010)\binom{4}{1\,3} \end{aligned}$$

$$
\begin{aligned}
& +r(1101) \\
= \; & 43\binom{6}{1\,3\,2} + 11\binom{4}{1\,3} + 3 = 2627.
\end{aligned}
$$

2.5 Randomizing Functions on a Prefix-Free Domain

The process of coin flipping is well represented by a complete binary tree; by the result of each coin flip we decide whether to flip another coin or to stop and output. In this section, we study the functions defined on exhaustive prefix-free sets over $\{H, T\}$ or, more generally over $\{0, 1, \ldots, s-1\}$. Since $\{H, T\}^n$ or $\{0, 1, \ldots, s-1\}^n$ are exhaustive prefix-free sets, the discussions in the previous sections are special cases. As an example, consider a function f that is represented by the following tree:



(2.20)

Let $U = B_0 \cup B_1$, where $B_0 = \{H\} \cup \{TH^{2k+1}T \mid k = 0, 1, 2, \ldots\}$ and $B_1 = \{TH^{2k}T \mid k = 0, 1, 2, \ldots\}$, and f outputs `0` and `1` on B_0 and B_1, respectively. If $p = \Pr[H] = 1/3$, then it is easy to verify that the probability that f outputs 0 is $1/2$. Hence f converts the probability distribution $\langle 1/3, 2/3\rangle$ to $\langle 1/2, 1/2\rangle$. However, if the bias p is not $1/3$, then the probability that f outputs 0 is not $1/2$.

Let U be an exhaustive prefix-free set over $\{0, 1, \ldots, s-1\}$. Let us call function $f : U \to \{0, 1, \ldots, t-1\}^*$ a *tree function*. If the set U is computable, then we can compute the corresponding function f. In this case, we may call such an algorithm *tree algorithm*. For example, in (2.20), f is computable using a finite automaton. In general, a prefix-free set may not be computable.

Call the probability distribution of the output of a tree function the *target distribution*. When the probability distribution of the source and the target of f are $\mathbf{p} = \langle p_0, \ldots, p_{s-1} \rangle$ and $\mathbf{r} = \langle r_0, \ldots, r_{t-1} \rangle$, respectively, let us say f is a tree function for $(\mathbf{p}, \mathbf{r})$. As in previous sections, the *efficiency* of a tree function is the average number of output target symbols per source symbol, and the *average cost* is its reciprocal. Then the efficiency of f is bounded above by the entropy bound $\mathcal{H}(\mathbf{r})/\mathcal{H}(\mathbf{p})$.

The target distribution of a *randomizing function* does not depend on the source distribution. We can formally define randomizing tree functions as in Section 2.3. By replacing $\{0, 1, \ldots, s-1\}^n$ by an exhaustive prefix-free set U over $\{0, 1, \ldots, s-1\}$ we obtain the formal definition. Also we can generalize E_s^n and $\tilde{E}_s^n$: Define $S_{\mathbf{d}}^U = U \cap S_{\mathbf{d}}$. Let $a_k m^k + a_{k-1} m^{k-1} + \cdots + a_0$ be the m-ary expansion of $|S_{\mathbf{d}}^U|$. Partition $S_{\mathbf{d}}^U$ into subsets A_i of size $a_i m^i$, where $0 < a_i \leq m$. Let E_s^U output a_i copies of $\{0, 1, \ldots, m-1\}^i$ on A_i. Then we can obtain $\tilde{E}_s^U$ as we obtained $\tilde{E}_s^n$ from E_s^n. Clearly, E_s^U and $\tilde{E}_s^U$ are randomizing. However, Theorem 2.16 does not generalize. (See the example (2.25) below.) Therefore, we cannot conclude that E_s^U and $\tilde{E}_s^U$ are optimal randomizing functions. Interestingly, we can see that the optimality of E_s^U and $\tilde{E}_s^U$ completely depends on U. We will give a precise condition that makes them optimal below. The discussion will be mostly restricted to binary case, and its generalization is straightforward.

In contrast to the randomizing tree functions, the optimality of tree functions for known bias are discussed in [26].

2.5.1 Linear Independence of Leaf Polynomials and Their Dimension

Let U be an exhaustive prefix-free set over $\{H, T\}$, and let $\mathcal{T}$ be the corresponding binary tree. The depth of the tree $\mathcal{T}$ is denoted by $\mathrm{depth}(\mathcal{T})$. For x in U, define $\mathcal{P}(x) = y^k(1-y)^l$,

where k and l are the number of the occurrences of H and T in x, respectively. Let us call $\mathcal{P}(x)$ the *leaf polynomial* of x. Also let us define $\mathcal{P}(\mathcal{T}) = \{\mathcal{P}(x) \,|\, x \in U\}$. A leaf polynomial can be seen as a real-valued function on $[0,1]$. Let $\dim(\mathcal{P}(\mathcal{T}))$ denote the dimension of the subspace spanned by $\mathcal{P}(\mathcal{T})$ of the vector space of real-valued functions on $[0,1]$. We are interested in $|\mathcal{P}(\mathcal{T})|$ and $\dim(\mathcal{P}(\mathcal{T}))$.

In the case of $U = \{H,T\}^n$, all the leaves of the corresponding tree $\mathcal{T}$ is of depth n, and clearly, $|\mathcal{P}(\mathcal{T})| = n+1$.

Proposition 2.19. *If $U = \{H,T\}^n$, then $\mathcal{P}(\mathcal{T})$ is linearly independent.*

Proof. Note that $\mathcal{P}(\mathcal{T}) = \{y^n, y^{n-1}(1-y), \ldots, (1-y)^n\}$. We need to show that if $a_n y^n + a_{n-1}y^{n-1}(1-y) + \cdots + a_0(1-y)^n = 0$, then $a_n = a_{n-1} = \cdots = a_0 = 0$. We will use the fact that $\{1, y, y^2, \ldots, y^n\}$ is linearly independent.

Note that

$$\begin{aligned} P(y) &= a_n y^n + a_{n-1}y^{n-1}(1-y) + \cdots + a_0(1-y)^n \\ &= [a_n - a_{n-1} + a_{n-2} - \cdots + (-1)^n a_0]y^n \\ &+ [a_{n-1} - 2a_{n-2} + 3a_{n-3} - \cdots + (-1)^{n-1} n a_0]y^{n-1} \\ &+ \cdots \\ &+ a_0 \\ &= \sum_{l=0}^{n} \left(\sum_{k=0}^{n} \mu_{lk} a_{n-k} \right) y^l, \end{aligned}$$

where

$$\mu_{lk} = \begin{cases} 0 & \text{if } l < k \\ (-1)^{k-l-n} \binom{k}{k-l-n} & \text{otherwise.} \end{cases}$$

The coefficients $a_0, \ldots, a_n$ map to the coefficients for $\{1, y, y^2, \ldots, y^n\}$ by a nonsingular linear mapping, whose matrix representation is $M = [\mu_{ij}]$. Note that M is upper triangular. Hence, $\mathcal{P}(\mathcal{T})$ is linearly independent, and $\dim(\mathcal{P}(\mathcal{T})) = n+1$. □

s	$\sigma(s,n), n=0,\cdots,7$							
0	1	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8
2	1	3	6	10	15	21	28	36
3	1	4	10	20	35	56	84	120
4	1	5	15	35	70	126	210	330
5	1	6	21	56	126	252	462	792

Table 2.4 Table of $\sigma(s,n)$

As a corollary of Proposition 2.19, if $U=\{H,T\}^n$, then $|\mathcal{P}(\mathcal{T})|=\dim(\mathcal{P}(\mathcal{T}))=n+1$.

Let $\sigma(s,n)$ be the number of the points $\mathbf{v}=(v_1,v_2,\ldots,v_s)$ in $\mathbf{N}^s$ such that $v_1+v_2+\cdots+v_s=n$, where $\mathbf{N}=\{0,1,2,\ldots\}$ is the set of natural numbers. If $U=\{0,1,\ldots,s-1\}^n$, then $|\mathcal{P}(\mathcal{T})|=\sigma(s,n)$. The function $\sigma(s,n)$ is defined recursively by $\sigma(s,n)=\sum_{k=0}^{n}\sigma(s-1,k)$ and $\sigma(0,n)=1$. Table 2.4 is the table of first a few values of $\sigma(s,n)$. We can see that $\sigma(s,n)=\binom{n+s}{s}$.

The linear independence of $\mathcal{P}(\mathcal{T})$ is most easily illustrated by a simple example.

Example Consider the case where $n=2, s=3$.

$$\begin{aligned}
P(x,y) &= ax^2+by^2+cxy+dx(1-x-y)+ey(1-x-y)+f(1-x-y)^2\\
&= [a-d+f]x^2\\
&+ [b-e+f]y^2\\
&+ [c-d-e+2f]xy\\
&+ [d-2f]x\\
&+ [e-2f]y\\
&+ f\cdot 1
\end{aligned}$$

Let a', b', c', d', e', and f' are the coefficients for the monomials x^2, y^2, xy, x, y, and 1, respectively. Then, in a matrix form, the two sets of coefficients are related as follows:

$$\begin{bmatrix} a' \\ b' \\ c' \\ d' \\ e' \\ f' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & -1 & 2 \\ 0 & 0 & 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} \tag{2.21}$$

Since the set of monomials $\{x^2, y^2, xy, x, y, 1\}$ is linearly independent, if $P(x, y) = 0$, then $a' = b' = c' = d' = e' = f' = 0$. Since the matrix in (2.21) is upper triangular, we conclude that $a = b = c = d = e = f = 0$. Hence the set of leaf polynomials for $U = \{0, 1, 2\}^2$ is linearly independent. The proof of the following proposition is essentially a formalization of the argument in this example.

Proposition 2.20. *If $U = \{0, 1, \ldots, s-1\}^n$, then $\mathcal{P}(\mathcal{T})$ is linearly independent.*

Proof. We use the linear independence of the set of monomials in $y_1, \ldots, y_{s-1}$ of degree at most n,

$$Q = \{y_1^{k_1} y_2^{k_2} \cdots y_{s-1}^{k_{s-1}} \mid 0 \leq k_1 + \cdots + k_s \leq n\}.$$

There are $\sigma(s, n) = \sigma(s-1, 0) + \cdots + \sigma(s-1, n)$ such monomials; there are $\sigma(s-1, k)$ monomials of degree k in Q. A polynomial in $y_1, \ldots, y_{s-1}$ of degree at most n is a linear combination of elements in Q. Now consider a canonical correspondence $\beta : Q \to \mathcal{P}(\mathcal{T})$ given by

$$y_1^{k_1} \cdots y_{s-1}^{k_{s-1}} \mapsto y_1^{k_1} \cdots y_{s-1}^{k_{s-1}} (1 - y_1 - \cdots - y_{s-1})^{n-k_1-\cdots-k_{s-1}}.$$

For the sake of a notational convenience, let us write $y_1^{k_1} \cdots y_{s-1}^{k_{s-1}}$ as $\mathbf{y}^{\mathbf{k}}$, where $\mathbf{y}$ denotes $(y_1, \ldots, y_{s-1})$ and $\mathbf{k}$ denotes $(k_1, \ldots, k_{s-1})$. We need to show that if $\sum_{\mathbf{k}} a_{\mathbf{k}} \beta(\mathbf{y}^{\mathbf{k}}) = 0$, then $a_{\mathbf{k}} = 0$ for every $\mathbf{k}$.

Consider the mapping $\mu : \mathbf{a} \mapsto \mathbf{b}$ defined by the equation

$$\sum_{\mathbf{k}} b_{\mathbf{k}} \mathbf{y}^{\mathbf{k}} = \sum_{\mathbf{j}} a_{\mathbf{j}} \beta(\mathbf{y}^{\mathbf{j}}), \tag{2.22}$$

where $\mathbf{a}$ and $\mathbf{b}$ are the vectors of coefficients in (2.22). The mapping is, in fact, linear. With an appropriate ordering of Q, or equivalently, an ordering on the set

$$K = \{(k_1, \ldots, k_{s-1}) \,|\, 0 \le k_1 + \cdots + k_{s-1} \le n\},$$

we will show that the matrix representation of the mapping μ is upper triangular.

Let us consider the counter-lexicographical order on the set K. That is, $\mathbf{k} = (k_1, \ldots, k_{s-1}) \prec \mathbf{k}' = (k'_1, \ldots, k'_{s-1})$ if $k_l > k'_l$, where l is the smallest index such that $k_l \neq k'_l$.

Note that $\beta(\mathbf{y}^{\mathbf{j}})$ is the sum of monomials $\mathbf{y}^{\mathbf{i}}$ such that $\mathbf{i} \preceq \mathbf{j}$. Define β_{ij} by $\beta(\mathbf{y}^{\mathbf{j}}) = \sum_{\mathbf{i}} \beta_{\mathbf{ij}} \mathbf{y}^{\mathbf{i}}$. Then

$$\beta_{\mathbf{ij}} = 0 \quad \text{if } \mathbf{j} \prec \mathbf{i}. \tag{2.23}$$

For a given $\mathbf{k}$, from the equation (2.22), we have $b_{\mathbf{k}} \mathbf{y}^{\mathbf{k}} = \sum_{\mathbf{j}} a_{\mathbf{j}} \beta_{\mathbf{kj}} \mathbf{y}^{\mathbf{k}}$. So we have $b_{\mathbf{k}} = \sum_{\mathbf{j}} \beta_{\mathbf{kj}} a_{\mathbf{j}}$, and the matrix for the linear map μ is $[\beta_{\mathbf{ij}}]$. Because of (2.23), $[\beta_{\mathbf{ij}}]$ is upper triangular. □

As a corollary of Proposition 2.20, if $U = \{0, 1, \ldots, s-1\}^n$, then $|\mathcal{P}(\mathcal{T})| = \dim(\mathcal{P}(\mathcal{T})) = \sigma(s, n)$.

Lemma 2.21. *For an exhaustive prefix-free set U over $\{H, T\}$, $\dim(\mathcal{P}(\mathcal{T})) \le \mathrm{depth}(\mathcal{T}) + 1$.*

Proof. Let $d = \mathrm{depth}(\mathcal{T})$ and let $\mathcal{T}_d$ be the tree corresponding to the set $\{H, T\}^d$. Then $\mathcal{T}$ can be obtained from $\mathcal{T}_d$ by applying a sequence of merging operations that merges two leaves that are siblings to each other. We will see that the merging operation on a tree does not increase the dimension of the set of leaf polynomials. This observation and Proposition 2.19 together imply the lemma.

Suppose that we have a binary tree $\mathcal{T}_1$. Let yg and $(1-y)g$ be the leaf polynomials corresponding to two leaf siblings, and let $\mathcal{T}_1'$ be the tree we obtain after merging the two leaf siblings. By merging them, we obtain the polynomial g. Note that $\{yg, (1-y)g\}$ and $\{g\}$ are linearly dependent because

$$yg + (1-y)g = g. \tag{2.24}$$

If g is linearly dependent on the other leaf polynomials of $\mathcal{T}_1'$, then $\{yg, (1-y)g\}$ is linearly dependent on the other leaf polynomials of $\mathcal{T}_1$ because of (2.24). So the $\dim(\mathcal{P}(\mathcal{T}_1)) = \dim(\mathcal{P}(\mathcal{T}_1'))$ in this case. If, otherwise, g is linearly independent of the other leaf polynomials of $\mathcal{T}_1'$, then $\{yg, (1-y)g\}$ is linearly independent on the other leaf polynomials of $\mathcal{T}_1$, again because of (2.24). So in this case, the merging operation decreases the dimension be 1, that is, $\dim(\mathcal{P}(\mathcal{T}_1)) = \dim(\mathcal{P}(\mathcal{T}_1')) + 1$. □

Lemma 2.22. *Let $U = U_H \cup U_T$, where U_H (U_T, respectively) is the subset of strings with prefix H (T). Let $\mathcal{T}_L$ and $\mathcal{T}_R$ the trees corresponding to L and R, respectively. Then* $\dim(\mathcal{P}(\mathcal{T})) = \max\{\dim(\mathcal{P}(\mathcal{T}_L)), \dim(\mathcal{P}(\mathcal{T}_R))\} + 1$.

Proof. Note that $\mathcal{P}(\mathcal{T}) = y\mathcal{P}(\mathcal{T}_L) \cup (1-y)\mathcal{P}(\mathcal{T}_R)$, where $y\mathcal{P}(\mathcal{T}_L) = \{yg \,|\, g \in \mathcal{P}(\mathcal{T}_L)\}$ and $(1-y)\mathcal{P}(\mathcal{T}_R) = \{(1-y)g \,|\, g \in \mathcal{P}(\mathcal{T}_R)\}$. Without loss of generality, assume that $\dim(\mathcal{P}(\mathcal{T}_L)) \geq \dim(\mathcal{P}(\mathcal{T}_R))$. If $g \in y\mathcal{P}(\mathcal{T}_L)$, then $g(0) = 0$. There is at least one polynomial h in $(1-y)\mathcal{P}(\mathcal{T}_R)$ such that $h(0) = 1$: the rightmost path of T_R yields $(1-y)^k$ for some k. Since $y\mathcal{P}(\mathcal{T}_L) \cup \{h\}$ is linearly independent, the lemma follows. □

By Lemma 2.21 and Lemma 2.22, we obtain the following theorem.

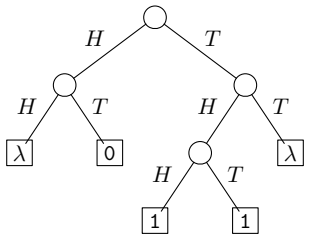
Theorem 2.23. *For an exhaustive prefix-free set over $\{H, T\}$,* $\dim(\mathcal{P}(\mathcal{T})) = \text{depth}(\mathcal{T}) + 1$.

2.5.2 Optimality of E_s^U and $\tilde{E}_s^U$

If $|\mathcal{P}(\mathcal{T})| = \dim(\mathcal{P}(\mathcal{T}))$, then the set of leaf polynomials is linearly independent. Since the proof of Theorem 2.3 depends on the linear independence of the leaf polynomials of $\{H, T\}^n$, if $|\mathcal{P}(\mathcal{T})| = \dim(\mathcal{P}(\mathcal{T}))$, then Theorem 2.3 holds for exhaustive prefix-free sets. As a consequence, the arguments in Section 2.2.3 generalize. So we have the following theorem.

Theorem 2.24. *Tree functions E_s^U and $\tilde{E}_s^U$ are optimal if $|\mathcal{P}(\mathcal{T})| = \dim(\mathcal{P}(\mathcal{T}))$.*

However, when $|\mathcal{P}(\mathcal{T})| > \dim(\mathcal{P}(\mathcal{T}))$, E_s^U may not be optimal. Consider the function f represented by the following tree.



(2.25)

The function f is randomizing on the corresponding prefix-free set

$$U = \{HH, HT, THH, THT, TT\}.$$

However, E_2^U does not generate a single output since $|S_{\mathbf{d}}^U| \leq 1$ for every $\mathbf{d}$. Therefore, E_2^U is not optimal. Now note that the depth of the tree (2.25) is 3. Hence the dimension of the corresponding set of leaf polynomials is 4. However, the tree has 5 different leaf polynomials.

Table 2.5 lists the number of binary trees that satisfy the condition $|\mathcal{P}(\mathcal{T})| = \dim(\mathcal{P}(\mathcal{T}))$. A *good tree* means the tree that satisfies the condition in Table 2.5. The table was computed using Theorem 2.23.

n	All complete binary trees	Good trees
2	1	1
3	2	2
4	5	5
5	14	12
6	42	29
7	132	70
8	429	169
9	1430	404
10	4862	969
11	16796	2318
12	58786	5544
13	208012	13246
14	742900	31660
15	2674440	75626

Table 2.5 The number of good trees

CHAPTER 3

The Optimality of Random Number Generation Using a Source of Known Distribution

In his doctoral thesis, Gauss showed that every algebraic equation has at least one root (Fundamental Theorem of Algebra). Abel, in 1828, went on to consider the same problem in a restricted model of computation. *He asked whether a root of every algebraic equation could be obtained using only arithmetic operations and the extraction of nth roots, and proved that the answer was negative. While all constructible numbers were known to be algebraic, this demonstrated that not all algebraic numbers are constructible. Shortly thereafter, he characterized those algebraic equations which* can *be solved by means of radicals, and this enabled him to discuss the feasibility of specific geometric problems, such as the trisection of the angle.*

— PREPARATA and SHAMOS, *in Computational Geometry, An Introduction (1985)*

It is difficult to say what is impossible, for the dream of yesterday is the hope of today and the reality of tomorrow.

— Robert H. Goddard

Since our main purpose is to establish the difficulty of simulation of a discrete probability distribution using a source of known distribution, in this chapter, we focus on the simplest case: simulating a coin of bias r using a coin of bias p.

3.1 DDG-Trees

3.1.1 DDG-trees and Random Number Generation

Knuth and Yao [20] represented their algorithms by binary trees. Consider a (possibly infinite) binary tree whose terminal nodes are labeled with `0` or `1`. Starting from the root node we perform a random walk based on coin flips. At each internal node, we take the left edge with *heads* (with probability p) and the right edge with *tails* (with probability q). If the random walk reaches a terminal node, then the associated label (`0` or `1`) is output. Suppose that r is the probability that `0` is output. If the random walk terminates with probability 1, then we can consider the binary tree as a description of an algorithm that converts coin flips (as a random source) with bias p to flips of another coin (as a target distribution) with bias r. We call such a binary tree a *discrete distribution generation tree (DDG-tree)* for a *configuration* (p, r). The target distribution generated by a DDG-tree algorithm will be written $\langle r, s\rangle$, where $s = 1 - r$.

The average depth of a DDG-tree T equals the average stopping time of the random walk on T, hence equals the average running time of the algorithm corresponding to T. But more important, it is also the average number of source coin flips to generate one target coin flip. A DDG-tree is *optimal* for a configuration (p, r), if there is no DDG-tree for (p, r) with a smaller average depth.

Figure 3.1 shows a DDG-tree for the configuration $(1/3, 1/2)$. Note that each subtree of a DDG-tree is a DDG-tree itself. We will often label an internal node of a DDG-tree with the target distribution of the subtree rooted at the node. The average depth of the DDG-tree is $15/7$. In fact, this DDG-tree is optimal for $(p, r) = (1/3, 1/2)$. (This fact will be proved in Section 3.3.3.) Can we construct an optimal DDG-tree for every (p, r)?

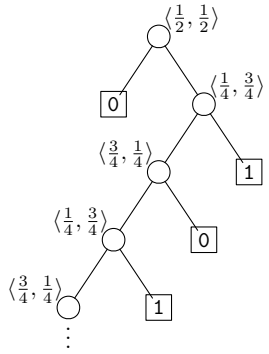


Figure 3.1 Simulating an unbiased coin using a coin of bias 1/3.

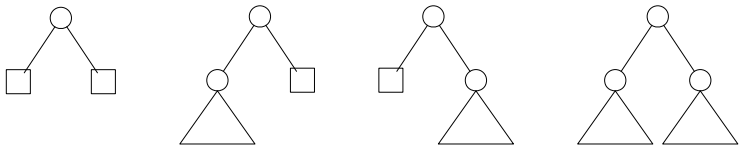
Throughout this paper, we assume that $p \leq 1/2$ is fixed and $q = 1 - p$, hence $p \leq q$. A left edge of a DDG-tree will always be associated with the *heads* of the source coin, which occurs with the probability p. For a DDG-tree T, let $E(T)$ denote the average depth of T. The root node is at level 0, and for a node v at level i, the children of v are at level $i + 1$.

3.1.2 Recursive Construction of DDG-Trees

In general, a DDG-tree is infinite. The target bias r of a DDG-tree is the sum of the terms of the form $p^i q^j$, and most values of r cannot be expressed by a finite sum of the terms $p^i q^j$. For example, there is no finite DDG-tree for configuration $(1/3, 1/2)$ because no finite sum of the terms $(1/3)^i(2/3)^j$ can be equal to $1/2$. Therefore, we propose a model of recursive construction of DDG-trees.

Definition 3.1. *Let us call* 0 *and* 1 trivial DDG-trees *with target distributions* $\langle 1, 0\rangle$ *and* $\langle 0, 1\rangle$*, respectively. A DDG-tree* T *is either a trivial DDG-tree or a pair* (T_1, T_2) *with the target distribution* $\langle r, s\rangle = \langle pr_1 + qr_2, ps_1 + qs_2\rangle$*, where* T_1 *and* T_2 *are DDG-trees, and* $\langle r_1, s_1\rangle$ *and* $\langle r_2, s_2\rangle$ *are target distributions of* T_1 *and* T_2*, respectively. We call* T_1 *and* T_2 *the* left subtree *and the* right subtree *of* T*, respectively.*

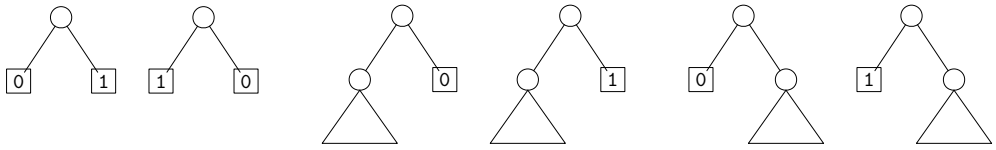
There are four possible *branchings* from an internal node:



(3.1)

An empty square denotes a trivial DDG-tree, which is a terminal node, and a triangle denotes a non-trivial DDG-tree. Let us call the four possible branchings $\mathtt{S}, \mathtt{L}, \mathtt{R}$, and $\mathtt{B}$, respectively, for *stop, left, right*, and *both*. The *type* of a DDG-tree is the branching at the root node, namely, $\mathtt{S}, \mathtt{L}, \mathtt{R}$, or $\mathtt{B}$. We say that an internal node *takes* a branching b if the branching at the internal node is b.

In the case of $\mathtt{S}, \mathtt{L}$, and $\mathtt{R}$, the branching includes a terminal node. By specifying the label for terminal node, we have the branchings shown below; let us call them $\mathtt{S0}, \mathtt{S1}, \mathtt{L0}, \mathtt{L1}, \mathtt{R0}$, and $\mathtt{R1}$, respectively.



(3.2)

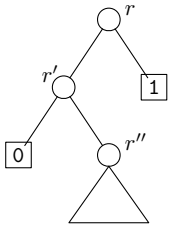
When branching $\mathtt{L}$ or $\mathtt{R}$ is taken, one of the subtrees is trivial. Since, by definition, the target distribution of a DDG-tree T is the weighted sum of the target distributions of the subtrees, $\langle r, s\rangle = \langle pr_1 + qr_2, ps_1 + qs_2\rangle$, and the target distribution of trivial subtree is constant, the target distribution of the non-trivial subtree is uniquely determined by the target distribution of T.

A *DDG-branching function* is a function of the form

$$F : (0, 1/2] \times (0, 1) \longrightarrow \{\mathtt{S0}, \mathtt{S1}, \mathtt{L0}, \mathtt{L1}, \mathtt{R0}, \mathtt{R1}, \mathtt{B}\} \times (0, 1).$$

A DDG-branching function F *induces* a unique DDG-tree for a given configuration (p, r) as follows. Suppose that $F(p, r) = (b, r')$. The branching at the internal node v with target bias r is specified by the value of b in $\{\mathtt{S0}, \mathtt{S1}, \mathtt{L0}, \mathtt{L1}, \mathtt{R0}, \mathtt{R1}, \mathtt{B}\}$. The target biases of subtrees of v are determined by r' in $(0, 1)$. If the branching b is $\mathtt{L}$ or $\mathtt{R}$, then the target

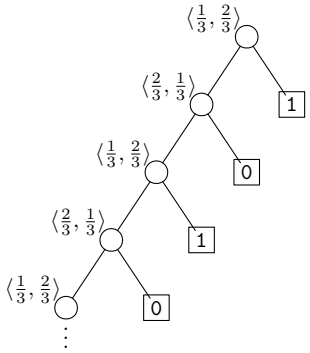
bias of the nontrivial subtree is r'. If $b = \texttt{B}$, then there are two non-trivial subtrees, and the target biases for them are $r_1 = r'$ and $r_2 = (r - pr_1)/q$, respectively. Then the subtrees are determined by F recursively. For example, if $F(p, r) = (\texttt{L1}, r')$, and $F(p, r') = (\texttt{R0}, r'')$, then the induced tree is the following.



Call a DDG-tree defined in this manner an *F-tree*, and call this procedure a *recursive construction* of DDG-tree.

3.1.3 Examples of Recursive Constructions of DDG-trees

The Knuth-Yao method [20] is based on the binary expansions of the target bias r and $s = 1 - r$: the Knuth-Yao DDG-tree has a depth-k terminal node with label `0` (`1`, respectively) if and only if the kth coefficient of the expansion of r (s) is 1. For example, suppose that we want to simulate the binary distribution $\langle 1/3, 2/3 \rangle$. Based on the binary expansions $1/3 = (0.010101\cdots)_2, 2/3 = (0.101010\cdots)_2$, the Knuth-Yao method results in the following DDG-tree for the configuration $(1/2, 1/3)$.



(3.3)

Although this DDG-tree has only branching `L` at each internal node, branching `R` can replace it because the source is unbiased. A Knuth-Yao DDG-tree is optimal because the

truncated tree of each finite depth gives the best approximation of the target distribution, as the binary expansion is the best possible approximation by finite sum of powers of 1/2. Although Knuth and Yao did not mention in their paper, thier method has a straightforward generalization for m-ary uniform source, for $m \geq 2$, and the generalization results in optimal simulations. The DDG-trees generated by Knuth-Yao method are induced recursively by the following DDG-branching function

$$F_{KY}(r) = \begin{cases} (\texttt{S0}, -) & \text{if } r = \frac{1}{2} \\ (\texttt{L0}, 2r) & \text{if } r < \frac{1}{2} \\ (\texttt{L1}, 2r-1) & \text{if } r > \frac{1}{2} \end{cases} \tag{3.4}$$

Since the source bias is fixed at 1/2, F_{KY} takes only r as an argument. The recursive construction by F_{KY} is essentially the same as the recursive computation of the standard binary expansion.

The Han-Hoshi method [13] handles an arbitrary source distribution. In the following discussion, we deal with the special case of Han-Hoshi method, where the source and the target are coins. The target distribution is represented as a partition of the unit interval, and the partition is approximated by a union of subintervals of the size $p^k q^l$. For example, Figure 3.2 shows the subdivision process for configuration $(1/3, 1/2)$. The dot on 1/2 divides the unit interval into two subintervals, $[0, 1/2)$ and $[1/2, 1]$, corresponding to the target distribution $\langle 1/2, 1/2 \rangle$. We call the subintervals *target subintervals*. Now we consider another subdivision into $[0, 1/3)$ and $[1/3, 1]$, which we call *approximating subintervals*. The ratio of approximating subintervals is always $p : q$. Since $[0, 1/3)$ is included in $[0, 1/2)$, we cross it out. At the second step, we do the same thing with the remaining interval $[1/3, 1]$, and now the target subintervals are $[1/3, 1/2)$ and $[1/2, 1]$. The approximating subintervals are $[1/3, 5/9)$ and $[5/9, 1]$. Since $[5/9, 1]$ is included in the target subinterval $[1/2, 1]$, we cross out $[5/9, 1]$. So at the third step, the target subintervals are $[1/3, 1/2)$ and $[1/2, 5/9)$, and we proceed in this manner. Note that

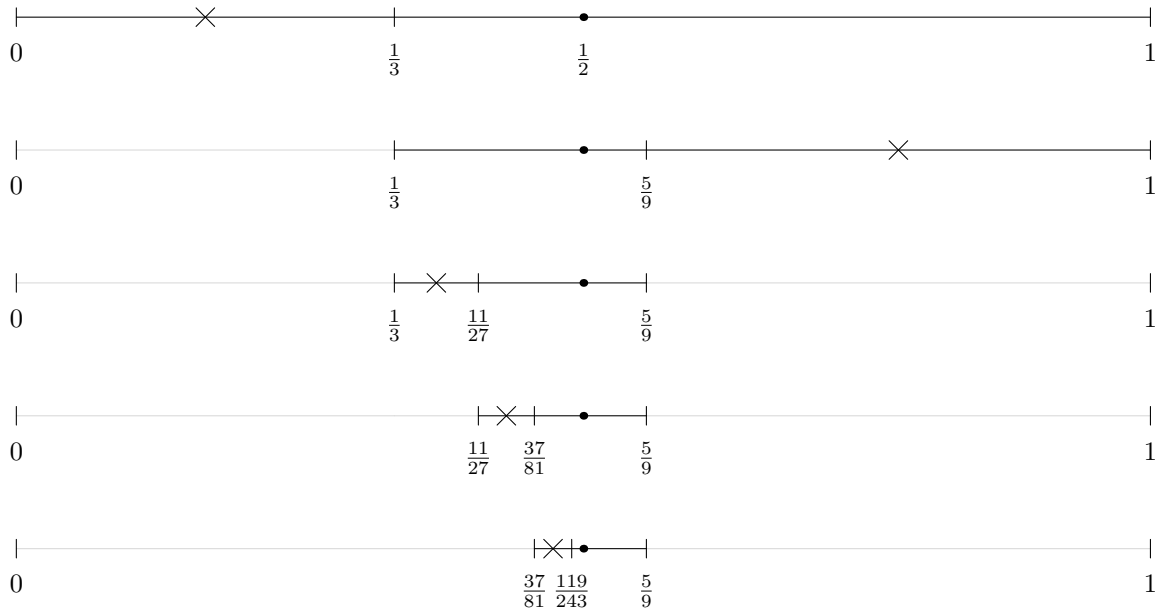
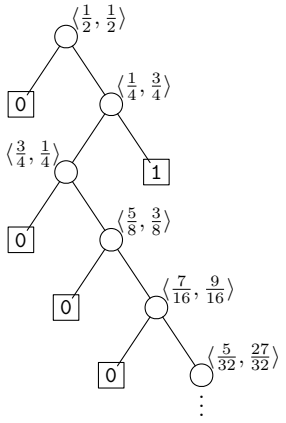


Figure 3.2 Han-Hoshi interval subdivisions

original target intervals are approximated by approximating intervals: $[0, 1/2) = [0, 1/3) \cup [1/3, 11/27) \cup [11/27, 37/81) \cup [37/81, 119/243) \cup \cdots$, and $[1/2, 1] = [5/9) \cup \cdots$.

This process corresponds to a DDG-tree. A crossed-out approximating subinterval corresponds to a terminal node: if it is in the left (right, respectively) target subinterval, then the output label is `0` (`1`). And the size of the crossed out approximating subinterval is the probability that the random walk reaches the corresponding terminal node. The sum of the sizes of the target intervals (the sizes of solid parts) over all steps is the average depth of the corresponding DGG-tree. The interval subdivision process in Figure 3.2

corresponds to the following DDG-tree.



(3.5)

The average depth of the DDG-tree is greater than 2.2, which is larger than the average depth of the DDG-tree in Figure 3.1 for the same configuration. So the DDG-tree (3.5) is not optimal. The DDG-trees generated by the Han-Hoshi algorithm are induced recursively by the DDG-branching function:

$$F_{HH}(p,r) = \begin{cases} (\texttt{S0}, -) & \text{if } r-p = 0 \\ (\texttt{L1}, r/p) & \text{if } r-p < 0 \\ (\texttt{R0}, (r-p)/q) & \text{if } r-p > 0. \end{cases} \tag{3.6}$$

Now we introduce a new method that we call A1. The DDG-tree in Figure 3.1 is constructed by A1. This method is based on the intuition that it is best to take branching `S` if possible, and it is likely better to take branching `L` than `R` because a right edge is heavier than a left edge at the same level. This idea can be explained in terms of interval subdivision. Figure 3.3 shows the interval subdivisions of A1 for the configuration $(1/3, 1/2)$. Note that, in the Han-Hoshi method, the lengths of approximating subintervals are always in ratio $p : q$. In A1, the ratio of approximating subintervals can be either in ratio $p : q$ or in ratio $q : p$. Compare Figure 3.2 and Figure 3.3. The first two steps are identical. But the third step of A1 divides the interval in the ratio $q : p$. By doing so, the interval in the fourth step is now smaller than the interval of Han-Hoshi.

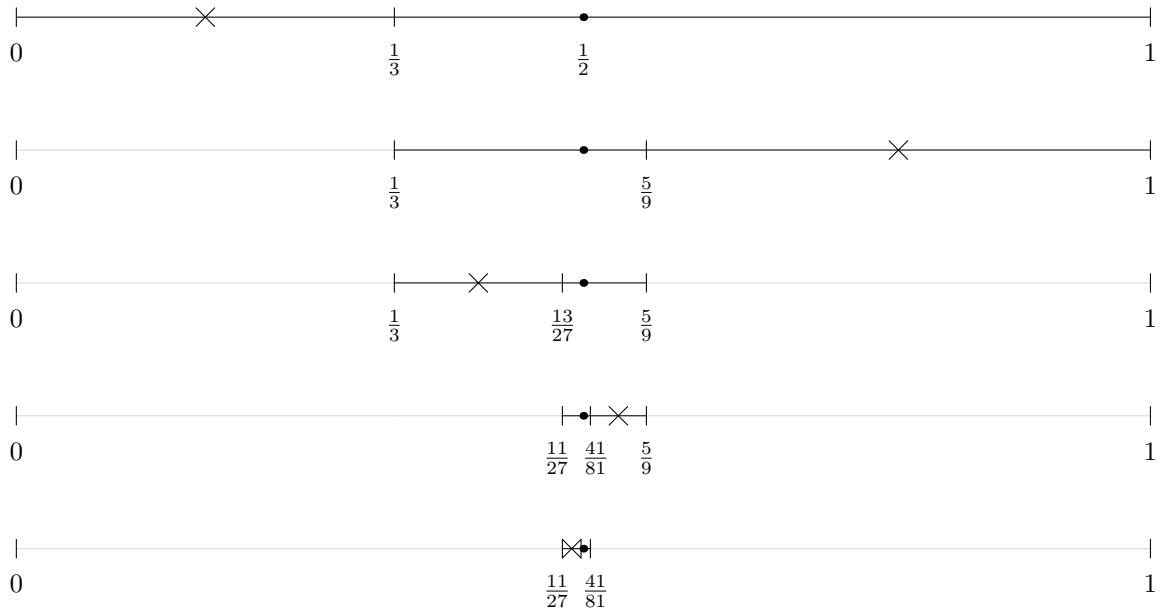


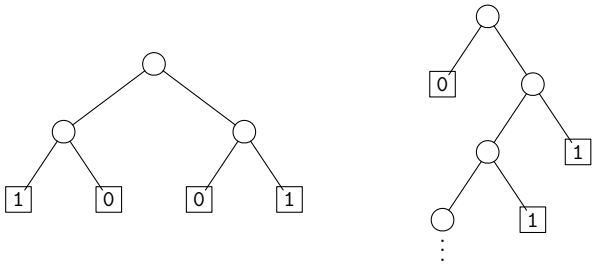
Figure 3.3 A1 interval subdivisions

For simplicity, call a DDG-tree constructed by A1 an A1-tree. The algorithm is described by the following DDG-branching function:

$$F_{A1}(p,r) = \begin{cases} (\texttt{S0}, -) & \text{if } r-p=0 \\ (\texttt{S1}, -) & \text{if } s-p=0 \\ (\texttt{L1}, r/p) & \text{if } r-p<0 \\ (\texttt{L0}, (r-q)/p) & \text{if } s-p<0 \\ (\texttt{R0}, (r-p)/q) & \text{if } r-p>0 \text{ and } r<s \\ (\texttt{R1}, r/q) & \text{if } s-p>0 \text{ and } r\geq s. \end{cases} \tag{3.7}$$

Both the Han-Hoshi method and A1 generalize the Knuth-Yao method: When $p = 1/2$, both methods generate essentially the same tree as the Knuth-Yao method. Therefore, both methods generate optimal DDG-trees for $p = 1/2$. But in general DDG-trees constructed by the methods are not optimal; we have seen that a Han-Hoshi DDG-tree

is not optimal earlier. Consider the DDG-trees (3.8) for $p = 0.1$ and $r = 2pq = 0.18$. The tree on the left has the average depth 2. The tree on the right is generated by A1, and its average depth is at least 2.071. Hence it is not an optimal DDG-tree.



(3.8)

In Section 3.3.3, we will prove that A1-trees are optimal for some configurations. As a corollary, we will obtain the optimality of the DDG-tree in Figure 3.1.

3.2 Impossibility of Recursive Construction of Optimal DDG-Trees

In this section, we address the computability of the recursive construction of an optimal DDG-tree. In order to argue about computability, we need a computation model.

3.2.1 Algebraic Computation Models and DDG-branching Functions

In Section 3.1.3, we showed that previously known methods for our problem can be seen as recursive constructions based on DDG-branching functions. Also we introduced a new method called A1 that is defined by a DDG-branching function. Observe that all the DDG-branching functions are expressed by simple algebraic expressions and decisions based on the values of simple algebraic formulas; whether $r - p$ is greater than 0, etc. Such a computaion is well-modeled by classical algebraic computation models.

The computational complexity of decision problems involving computations of real numbers is studied using computation models such as the algebraic decision tree [32], the algebraic computation tree [2] and the real-RAM. Let A be a subset of $\mathbf{R}^n$. A *decision problem* P on a domain A is a mapping from A to a finite set, say, $\{0, 1, \ldots, k\}$. Suppose that P is computed by an algebraic computation model. Then, for each i in $\{0, 1, \ldots, k\}$, $P^{-1}(i)$ is a set of solutions in A of a system of polynomial equations and inequalities that are derived by the corresponding computation path. A set of solutions of a system of polynomial equations and inequalities is called a *semi-algebraic set*. The number of connected components of a semi-algebraic set is finite [23, 25, 34]. The number of connected components of $P^{-1}(i)$ is used to determine the computational complexity of the decision problem P in terms of algebraic computation models (for example, the depth of an algebraic decision tree). More sophisticated topological invariants like Betti numbers are also used [36].

We prove that our problem requires an infinite number of connected components. Therefore, the problem is not solvable using classical algebraic computation models.

Given a DDG-branching function F, let $\tilde{F} : (0, 1/2] \times (0, 1) \longrightarrow \{\texttt{S0}, \texttt{S1}, \texttt{L0}, \texttt{L1}, \texttt{R0}, \texttt{R1}, \texttt{B}\}$ be the projection of F such that $\tilde{F}(p, r) = b$, where $F(p, r) = (b, r')$. The function $\tilde{F}$ makes a decision about the kind of branching of the DDG-tree that F induces. For all DDG-braching functions that we considered in Section 3.1.3, the corresponding projections can be computed by classical algebraic computation models. For example, Figure 3.4 shows $\tilde{F}_{HH}$, the projection of Han-Hoshi DDG-branching function.

3.2.2 Case $r = 2pq$

Consider the case where the target bias $r = 2pq$. Let $C_{2pq} = \{(p, r) \,|\, r = 2pq,\ 0 < p \leq 1/2\}$. The following tree is a DDG-tree for all configurations in C_{2pq}. Call this DDG-tree

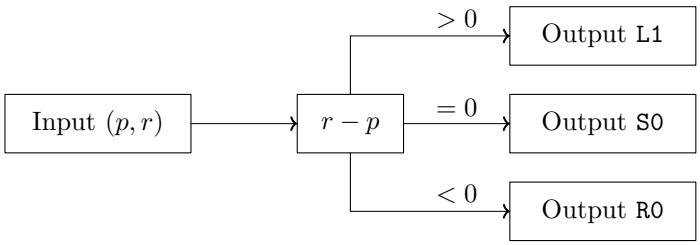
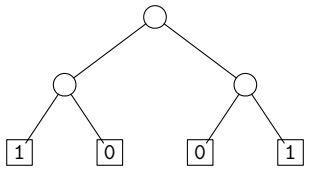


Figure 3.4 Projection of Han-Hoshi DDG-branching function.

T_{2pq}.



(3.9)

We will show that T_{2pq} is the *unique* optimal tree for configurations in a dense subset of C_{2pq}.

Suppose that F^* is a DDG-branching function that induces optimal DDG-trees. So, if (p, r) is a configuration in this subset, then F^* must induce T_{2pq} for (p, r) and $\tilde{F}^*(p, r) = \texttt{B}$. But we will also show that there are infinitely many configurations (p, r) in C_{2pq} for which T_{2pq} is not optimal, and that for those configurations $\tilde{F}^*$ takes R. Our main result, Theorem 3.3, is based on these facts.

Note that $E(T_{2pq}) = 2$ for every p in $(0, 1/2]$. Therefore if there is a DDG-tree $T \neq T_{2pq}$ for $(p, 2pq)$, for a fixed $p \in (0, 1/2]$, with an average depth at most 2, then T_{2pq} is not the unique optimal DDG-tree for the configuration $(p, 2pq)$. Let us define

$$\mathcal{A} = \{p \in (0, 1/2] \mid T_{2pq} \text{ is the unique optimal DDG-tree for } (p, 2pq).\}$$

$$\mathcal{B} = \{p \in (0, 1/2] \mid \text{There is an infinite DDG-tree } T \text{ for } (p, 2pq), \text{ and } E(T) = 2.\}$$

$$\mathcal{C} = \{p \in (0, 1/2] \mid \text{There is a DDG-tree } T \text{ for } (p, 2pq), \text{ and } E(T) < 2.\}$$

$$\mathcal{D} = \{p \in (0, 1/2] \mid \text{There is a finite DDG-tree } T \text{ for } (p, 2pq), T \neq T_{2pq} \text{ and } E(T) = 2.\}$$

Theorem 3.2. *The following hold for $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$.*

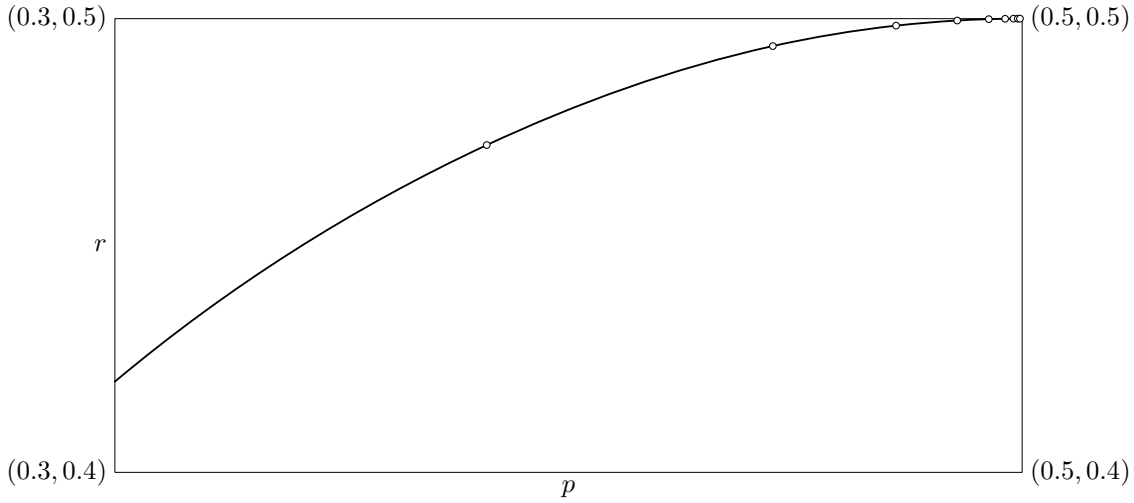


Figure 3.5 The set C_{2pq}, the configurations (p, r) such that $r = 2pq$. The small circles on the curve correspond to the configurations $(p, 2pq)$ for some samples of p in $\mathcal{C}$. The DDG-trees corresponding to the configurations are presented in Section 3.4.3.

(i) *$\mathcal{A}$ is a dense subset of $(0, 1/2]$.*

(ii) *$\mathcal{B}$ is uncountable and nowhere dense.*

(iii) *$\mathcal{C} \cup \mathcal{D}$ is countable, and $\mathcal{C}$ is infinite.*

(iv) *If a DDG-tree T for the configuration $(p, 2pq)$ is not T_{2pq}, and $E(T) \leq 2$, then T takes branching* `R` *at the root.*

Note that T_{2pq} is not the unique optimal DDG-tree for $(p, 2pq)$ if and only if there is a DDG-tree T for $(p, 2pq)$ such that $T \neq T_{2pq}$ and $E(T) \leq 2$. Therefore $\mathcal{A} = (0, 1/2] \setminus (\mathcal{B} \cup \mathcal{C} \cup \mathcal{D})$. Therefore, parts (ii) and (iii) of Theorem 3.2 together imply that $\mathcal{A}$ is dense in $(0, 1/2]$. The proofs of parts (ii), (iii) and (iv) of Theorem 3.2 are given in Section 3.4.

Theorem 3.3. *No algebraic computation tree can decide the branchings for a recursive construction of an optimal DDG-tree for all (p, r).*

Proof. Suppose to the contrary that there is a DDG-branching function F^* such that F^*-trees are optimal, and suppose $\tilde{F^*}$ can be computed by an algebraic computation tree. Let us consider the behavior of $\tilde{F^*}$ on C_{2pq}, the curve shown in Figure 3.5. If $p \in \mathcal{A}$, then $\tilde{F^*}(p, 2pq) = \mathtt{B}$ because for such p, T_{2pq} is the unique optimal DDG-tree and it takes the branching $\mathtt{B}$ at the root. Similarly, if $p \in \mathcal{C}$, then T_{2pq} is not optimal and $\tilde{F^*}(p, 2pq) = \mathtt{R}$ by (iv) of Theorem 3.2. If $p \in (\mathcal{B} \cup \mathcal{D}) \setminus \mathcal{C}$, then a DDG-tree of average depth 2 is optimal, and it can be either T_{2pq} or a DDG-tree of type $\mathtt{R}$. So in this case, $\tilde{F^*}(p, 2pq)$ may be either $\mathtt{B}$ or $\mathtt{R}$. Therefore, we can conclude that

$$\mathcal{A} \subset \{p \,|\, \tilde{F^*}(p, 2pq) = \mathtt{B}\} \subset \mathcal{A} \cup ((\mathcal{B} \cup \mathcal{D}) \setminus \mathcal{C}) = (0, 1/2] \setminus \mathcal{C}. \tag{3.10}$$

The last equality in (3.10) holds because $\mathcal{A} \cap \mathcal{C} = \emptyset$, and $\mathcal{A} \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{D} = (0, 1/2]$.

Let $C = \tilde{F^*}^{-1}(\mathtt{B}) \cap C_{2pq}$. Since F^* is computed by an algebraic computation tree, $\tilde{F^*}^{-1}(\mathtt{B})$ is a semi-algebraic set. Clearly C_{2pq} is a semi-algebraic set too. Therefore, C has a finite number of connected components.

Consider the projection $C' = \{p \,|\, (p, r) \in C\}$, which is a homeomorphic image of C. Note that $C' = \{p \,|\, \tilde{F^*}(p, 2pq) = \mathtt{B}\}$, and by (3.10) we have $\mathcal{A} \subset C' \subset (0, 1/2] \setminus \mathcal{C}$. By Theorem 3.2, $\mathcal{A}$ is dense, and therefore C' is a dense subset of $(0, 1/2]$. Also C' does not contain any point in $\mathcal{C}$, and $\mathcal{C}$ is an infinite set by (iv) of Theorem 3.2. Therefore C' has an infinite number of connected components, a contradiction. □

3.3 Optimal Infinite DDG-Trees

We show that A1-trees are optimal for the configurations for which no finite DDG-tree exists and the source bias is close to 1/2. First, we prove some upper bounds for A1-trees and lower bounds for general DDG-trees.

Remember that the *type* of a DDG-tree is the branching at the root node. We say that an internal node *takes* a branching b if the branching at the internal node is b. We also say that a DDG-tree T takes a branching b at level i, if an internal node ith level takes branching b. Since an A1-tree takes only branchings $\mathtt{S}, \mathtt{L}$ or $\mathtt{R}$, there is a unique internal node at each level.

3.3.1 Upper Bounds for A1-Trees

We analyze the average depth of A1-trees. Call a path on a DDG-tree that is obtained by taking consecutive right branchings a *right branching path*. Define $K(p, r)$ to be the length of the first right branching path starting at the root of the A1-tree for the configuration (p, r). By definition, if the type of A1-tree for the configuration (p, r) is not $\mathtt{R}$, then $K(p, r) = 0$.

Lemma 3.4. *For a configuration* (p, r), $K(p, r) = \lceil \log_q \max(r, s) \rceil - 1$.

Proof. Let T be the A1-tree for (p, r). We proceed by induction on the value of $K(p, r)$. Suppose that $K(p, r) = 0$. By definition, the type of T is not $\mathtt{R}$. Hence $\max(r, s) \geq q$. (See the definition of A1 (3.7).) So $\lceil \log_q \max(r, s) \rceil - 1 = 0$, and the lemma holds for $K(p, r) = 0$.

Suppose that $K(p, r) = n \geq 1$. Since the type of T is $\mathtt{R}$, the target distribution for the internal node at level 1 is $\langle r', s' \rangle = \langle (\min(r, s) - p)/q, \max(r, s)/q \rangle$. In the subtree with the target distribution $\langle r', s' \rangle$, the length of the first right branching path of length is $n - 1$. By the induction hypothesis, $K(p, (\min(r, s) - p)/q) = \lceil \log_q(\max(r, s)/q) \rceil - 1 = n - 1$. Therefore, $\lceil \log_q \max(r, s) \rceil - 1 = n$, which completes the proof. □

Note that $K(p, r)$ is the minimum k such that $\max(r, s) > q^{k+1}$. Lemma 3.4 implies that a right branching path of an A1-tree cannot be infinite. Let $K^*(p) = K(p, 1/2)$. For a

fixed p, $K(p,r)$ is bounded above by $K^*(p) = \lceil \log_q \frac{1}{2} \rceil - 1$. Therefore, the length of a right branching path, not necessarily the first one, in an A1-tree is at most $K^*(p)$. Let us write K and K^*, respectively, for $K(p,r)$ and $K^*(p)$ where the context is clear. Let $U(p,r)$ be the average depth of the A1-tree for configuration (p,r) and $U^*(p) = \sup_{0 \le r \le 1/2} U(p,r)$. Then

$$\begin{aligned} U^*(p) &\le p + 2pq + 3pq^2 + \cdots + K^* pq^{K^*-1} + (K^*+1)q^{K^*+1} + pq^{K^*}\left(K^* + 1 + U^*(p)\right) \\ &= p + 2pq + 3pq^2 + \cdots + K^* pq^{K^*-1} + (K^*+1)pq^{K^*} + (K^*+1)q^{K^*+1} + pq^{K^*}U^*(p) \\ &= \frac{1 - q^{K^*+1}}{p} - (K^*+1)q^{K^*+1} + (K^*+1)q^{K^*+1} + pq^{K^*}U^*(p) \\ &= \left(1 + q + \cdots + q^{K^*}\right) \Big/ (1 - pq^{K^*}). \end{aligned}$$

Theorem 3.5. *The average depth of an A1-tree is at most*

$$\left(1 + q + \cdots + q^{K^*(p)}\right) \Big/ (1 - pq^{K^*(p)}).$$

This upper bound is sharp for p such that $K^*(p) = 1, 2, \cdots$. When $K^*(p) = 1$, or equivalently, when $1 - 1/\sqrt{2} \le p < 1/2$, we have $U^*(p) \le (1+q)/(1-pq)$.

Observe that the bound in Theorem 3.5 is independent of the target distribution $\langle r, s \rangle$.

Lemma 3.6. *Let T_L and T_R be DDG-trees of types* `L` *and* `R`*, respectively. The following statements about the average depth of DDG-trees hold.*

(UB1) *If T_L is an A1-tree, then $E(T_L) < 2$.*

(UB2) *If T_R is an A1-tree, and $p \ge 0.32$, then $E(T_R) \le U^*(p) \le (1+q)/(1-pq)$.*

Proof. (UB1) Let T' be the subtree rooted at the internal node at the level 1. Since T' is an A1-tree, $E(T') \le U^*(p)$. Note that $U^*(p) < 1/p$. Hence $E(T_L) = 1 + pE(T') < 2$.

(UB2) Since $p \geq 0.32 > 1 - 1/\sqrt{2}$, $K^*(p) = 1$. Since T_R is an A1-tree, $E(T_R) \leq U^*(p) \leq (1+q)/(1-pq)$. □

3.3.2 Lower Bounds

The following lemma states lower bounds on the average depth of a DDG-tree depending on the branching at the root.

Lemma 3.7. *Let T_S, T_L, T_R and T_B be DDG-trees of types* `S`, `L`, `R` *and* `B`, *respectively. The following statements about the average depth of DDG-trees hold.*

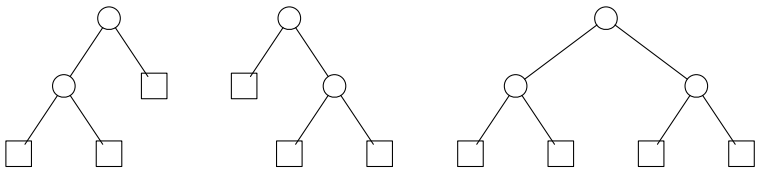
(LB1) $E(T_S) = 1$, $E(T_L) \geq 1 + p$, $E(T_R) \geq 1 + q$, *and* $E(T_B) \geq 2$.

(LB2) *If T is an infinite DDG-tree, then $E(T) \geq 1/q$. If $p \neq 1/2$ and T is an infinite DDG-tree of average depth $1/q$, then every internal node of T takes branching* `L`.

(LB3) *If T_R is infinite, then $E(T_R) \geq 2$. If $p \neq 1/2$ and T_R is an infinite DDG-tree of average depth 2, then every internal node of T_R, except the root node, takes branching* `L`.

(LB4) *If T_B is infinite, then $E(T_B) \geq 2 + p^2/q$.*

Proof. (LB1) It is clear that $E(T_S) = 1$. The following are the DDG-trees with the smallest average depths of type `L`, `R`, and `B`, and their average depths are $1+p$, $1+q$, and 2, respectively.



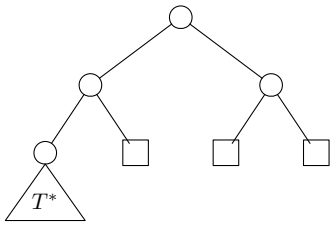
(LB2) Let T^* be a DDG-tree that takes a left branching at every internal node. Then $E(T^*) = 1 + p + p^2 + \cdots = 1/q$. Suppose that T is an infinite DDG-tree such that

$E(T) \leq 1/q$. We will show that every internal node takes branching `L`. Hence, $1/q$ is the tight lower bound, and $T = T^*$.

First, we show that any internal node of T cannot take branching `B`. Suppose that T takes a first branching `B` at level k; T takes only `L` or `R` at each node at ith level, for $i < k$. Let T' be the subtree rooted at the kth internal node. Note that $E(T) = 1 + \sum_{i=1}^{k-1} p^{d_i} q^{i-d_i} + p^{d_k} q^{k-d_k} E(T')$, where $\{d_i\}$ is monotonically increasing sequence such that $e_i = d_{i+1} - d_i$ is 0 or 1. The ith internal node takes branching `L` if and only if $e_i = 1$. By (LB1), $E(T') \geq 2$. Since $p \leq q$, we have $p^i \leq p^{d_i} q^{i-d_i}$, hence $E(T) \geq 1 + p + \cdots + p^k E(T') \geq 1 + p + \cdots + 2p^k > 1/q$, a contradiction. Hence we can assume that each internal node of T takes only `L` or `R`. For some monotonically increasing sequence $\{d_i\}$ as above, $E(T) = 1 + \sum_{i=1}^{\infty} p^{d_i} q^{i-d_i} \geq 1 + \sum_{i=1}^{\infty} p^i = 1/q$, where $\{d_i\}$ is as above. Note that if an internal node of T takes branching `R`, then $E(T)$ is strictly greater than $1/q$, unless $p = q = 1/2$.

(LB3) Let T' be the subtree rooted at the internal node at level 1. Then, by (LB2), $E(T_R) = 1 + qE(T') \geq 2$.

(LB4) Consider the following DDG-tree in which the DDG-tree T^* takes branching `L` at every internal node, as in the proof of (LB2).



Then the shown DDG-tree is an infinite DDG-tree of the smallest average depth with branching `B` at the root. □

3.3.3 Optimality for Infinite Trees

There are configurations (p, r) for which the only DDG-trees are infinite. For example, if p and r are rational, and $p = a/b$, $r = c/d$, in their lowest terms, and $\gcd(b, d) = 1$, then a DDG-tree for the configuration must be infinite because any finite tree will result in a rational target whose denominator is not prime to b. Also, if p is an irrational number (like Abrahams's algebraic coin [1]), and the target bias r is a rational number, then every DDG-tree for (p, r) is infinite.

Theorem 3.8. *If there is no finite DDG-tree for a configuration (p, r), and if $p \geq 0.32$, then the A1-tree for (p, r) is optimal.*

Proof. Let T be the A1-tree for (p, r), and let T' be a different DDG-tree for the same configuration. Suppose that the internal node of T at level m is the first to take a different branching from T'. Let T_m and T'_m are subtrees rooted at mth nodes of T and T', respectively. We will show that $E(T_m) \leq E(T'_m)$, which implies that $E(T) \leq E(T')$.

Let $\langle r_m, s_m \rangle$ be the target distribution for T_m. Evidently, $\langle r_m, s_m \rangle$ is also the target distribution for T'_m. Note that T_m and T'_m are infinite trees, and T_m is an A1-tree.

Case $p = r_m$ or $p = s_m$: Then $E(T_m) = 1 \leq E(T'_m)$ by (LB1) of Lemma 3.7.

Case $r_m < p$ or $s_m < p$: In this case, `L`, `R`, and `B` can be chosen. Since T_m is A1-tree, it takes `L` at the root node and thus $E(T_m) < 2$ by (UB1) of Lemma 3.6. By (LB1) and (LB3), $E(T'_m) \geq 2$. Hence $E(T_m) \leq E(T'_m)$.

Case $p < \min(r_m, s_m)$: In this case, `R` and `B` can be chosen. Since T_m is A1-tree, it takes `R` at the root node; if $r_m < s_m$, then `R0`, otherwise, `R1`. By (UB2), $E(T_m) \leq (1 + q)/(1 - pq)$. Without loss of generality suppose that $r_m < s_m$; so T_m takes `R0`. There are two choices for T'_m. If it takes `B`, then by (LB4), $E(T'_m) \geq 2 + p^2/q$. Since $(1 + q)/(1 - pq) \leq 2 + p^2/q$ for $p \geq 0.32$, we have $E(T_m) \leq (T'_m)$. So we only need to consider the case where T'_m chooses `R1`. We can assume that T'_m does not have branching `B`

by the above discussion. By Lemma 3.9, the first right branching path of T_m is not longer than that of T'_m. So T_m takes branching `L` no later than T'_m, hence $E(T_m) \leq E(T'_m)$. □

Lemma 3.9. *Suppose that T is an infinite DDG-tree for a configuration (p, r) that does not have branching* `B`*. Then the length of the first right branching path is at least $K(p, r)$.*

Proof. Suppose that the length k of the first right branching path of T is less than $K(p, r) = \lceil \log_q \max(r, s) \rceil - 1$. Then the internal node at level k takes `L`. Let $\langle r_k, s_k \rangle$ be the target distribution at this node. Let $\ell_i, i = 1, \ldots, k$ be the labels at the terminal nodes at level i. Then we have

$$\begin{aligned} r_k &= \frac{1}{q^k}\left(r - \sum_{i=1}^{k} \chi_{\mathtt{0}}(\ell_i) p q^{i-1}\right) \\ s_k &= \frac{1}{q^k}\left(s - \sum_{i=1}^{k} \chi_{\mathtt{1}}(\ell_i) p q^{i-1}\right) \end{aligned}$$

where $\chi_{\mathtt{0}}$ (respectively $\chi_{\mathtt{1}}$) is the characteristic function of `0` (`1`). Since the internal node at level k takes `L`, we have $\max(r_k, s_k) > q$. So either $r - \sum_{i=1}^{k} \chi_{\mathtt{0}}(\ell_i) p q^{i-1} > q^{k+1}$ or $s - \sum_{i=1}^{k} \chi_{\mathtt{1}}(\ell_i) p q^{i-1} > q^{k+1}$ holds. However, $\max(r, s) < q^{K(p,r)} < q^k$, a contradiction. □

The DDG-tree shown in Figure 3.1 is an A1-tree. There is no finite DDG-tree for the configuration $(1/3, 1/2)$ because the denominators 3 and 2 are relatively prime, and $p = 1/3 > 0.32$. Hence the tree in Figure 3.1 is optimal.

For a fixed p, at most countably many values of r allow finite trees. Therefore, for a fixed $p \geq 0.32$, algorithm A1 constructs an optimal DDG-tree for almost every r.

3.4 Proof of Theorem 3.2

In this section, we prove Theorem 3.2 by characterizing the optimal DDG-trees for the configurations in $C_{2pq} = \{(p, r) \mid r = 2pq,\ 0 < p \leq 1/2\}$. First we prove the assertion (iv).

Consider the possible branchings for the target $r = 2pq$, and consider the DDG-trees with average depth at most 2. When $p = 1/2$, of course, $r = 2pq = 1/2$, and the two trees of type `S`, whose average depths are 1, are optimal. We exclude this case and assume $p < 1/2$. Since $p < r < s$, the branchings `S` and `L` are not possible. The average depth of the trees of type `B` is at least 2 (Lemma 3.7, (LB1)), and the only tree of this type with average depth at most 2 is T_{2pq}. Therefore we proved assertion (iv) of Theorem 3.2 and put it as a lemma.

Lemma 3.10. *If a DDG-tree T for the configuration $(p, 2pq)$ is not T_{2pq}, and $E(T) \leq 2$, then T takes branching* `R` *at the root.*

Now suppose that T is a DDG-tree of type `R` for the configuration $(p, 2pq)$ with average depth at most 2. If T is an infinite DDG-tree, then by Lemma 3.7, (LB3), $E(T) \geq 2$. So we conclude:

Lemma 3.11. *If T is an infinite DDG-tree for $(p, 2pq)$ and $E(T) \leq 2$, then $E(T) = 2$.*

Therefore, if $E(T) < 2$, then T must be a finite DDG-tree.

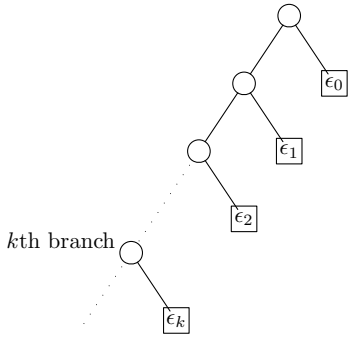
Lemma 3.12. *If T is DDG-tree for $(p, 2pq)$ and $E(T) < 2$, then T is finite.*

We first discuss the case where T is infinite. In order to do so, we discuss a special class of infinite DDG-trees in Section 3.4.1. Then we give a complete characterization of the case where T is infinite in Section 3.4.2. It seems rather difficult to give a full description of the case where T is finite, about which we will discuss in Section 3.4.3. However, we do not need a complete characterization of the case to prove our main result Theorem 3.3.

3.4.1 Preliminaries

In this section we discuss a class of infinite DDG-trees which has a one-to-one correspondence with the set of infinite binary sequences $\{\mathtt{0},\mathtt{1}\}^\omega$.

Given a sequence $\epsilon = \epsilon_0\epsilon_1\epsilon_2\cdots$ in $\{\mathtt{0},\mathtt{1}\}^\omega$, let T_ϵ be the infinite DDG-tree such that every internal node takes branching L, and ϵ_{i-1} is the label for ith terminal node:



(3.11)

Let $A_0(\epsilon) = \{i \mid \epsilon_i = 0\}$, and let $A_1(\epsilon) = \{i \mid \epsilon_i = 1\}$. Then the target distribution of T_ϵ is $\langle f_\epsilon(p), 1 - f_\epsilon(p)\rangle$, where $f_\epsilon(p) = \sum_{i\in A_0(\epsilon)} qp^i$.

Let $f_\epsilon(t) = \sum_{i\in A_0(\epsilon)}(1-t)t^i$, where $0 \le t \le 1/2$. Each summand is monotone; if $i = 0$, then decreasing, otherwise increasing. Because of the monotonicity of the summands, the derivative of f_ϵ is the sum of the derivatives of the summands of the sum that defines f_ϵ. It is straightforward to show that if $\epsilon_0 = \mathtt{0}$, then $f'_\epsilon(t) \le 0$ for $0 \le t \le 1/2$, hence f_ϵ is decreasing. If $\epsilon_0 = \mathtt{1}$, then each summand $(1-t)t^i$ is increasing and so is f_ϵ. So we conclude the following.

Lemma 3.13. *If $\epsilon_0 = \mathtt{0}$, then $f_\epsilon(0) = 1$, and f_ϵ is monotone decreasing. If $\epsilon_0 = \mathtt{1}$, then $f_\epsilon(0) = 0$, and f_ϵ is monotone increasing.*

Let us consider the strict lexicographical order $\prec$ on $\{\mathtt{0},\mathtt{1}\}^\omega$ based on the relation $\mathtt{0} \prec \mathtt{1}$. For example, $\mathtt{01}^\omega \prec \mathtt{10}^\omega$. If $\epsilon \prec \epsilon'$ and there is no sequence strictly between ϵ and ϵ', then we call ϵ the *immediate predecessor* of ϵ' and ϵ' the immediate successor of ϵ.

Lemma 3.14. *A sequence ϵ is an immediate predecessor if and only if $\epsilon = \delta\mathtt{01}^\omega$ for some $\delta \in \{\mathtt{0},\mathtt{1}\}^*$. In this case, the immediate successor of ϵ is $\delta\mathtt{10}^\omega$.*

Lemma 3.15. *If $\epsilon \prec \epsilon'$, then $f_\epsilon(t) > f_{\epsilon'}(t)$ for $0 < t < 1/2$.*

Proof. Suppose that $\epsilon = \epsilon_0\epsilon_1\cdots$, $\epsilon' = \epsilon'_0\epsilon'_1\cdots$, and that ϵ_k and ϵ'_k are the first symbols at which ϵ and ϵ' differ so that $\epsilon_k = \mathtt{0}$ and $\epsilon'_k = \mathtt{1}$. Let

$$f(t) = \sum_{i\in A_0(\epsilon), i\le k} (1-t)t^i.$$

Since each $(1-t)t^j$ is positive, for $0 < t < 1/2$, and $f_\epsilon(t)$ contains all the summands of $f(t)$, we have $f_\epsilon(t) \ge f(t)$ for $0 < t < 1/2$. Note that

$$t^k > \sum_{i=k+1}^{\infty} t^i = \frac{t^{k+1}}{1-t}$$

for $0 < t < 1/2$, since $1 - t > t$. Also note that

$$f_{\epsilon'}(t) = f(t) - (1-t)t^k + \sum_{i\in A_0(\epsilon'), i\ge k+1} (1-t)t^i.$$

So

$$\begin{aligned} f(t) - f_{\epsilon'}(t) &= (1-t)t^k - \sum_{i\in A_0(\epsilon'), i\ge k+1} (1-t)t^i \\ &> (1-t)t^k - \sum_{i=k+1}^{\infty} (1-t)t^i \\ &= (1-t)\left(t^k - \sum_{i=k+1}^{\infty} t^i\right) > 0 \end{aligned}$$

for $0 < t < 1/2$. Hence $f_\epsilon(t) \ge f(t) > f_{\epsilon'}(t)$ for $0 < t < 1/2$. □

Note that Lemma 3.15 does not hold for $t = 1/2$, in which case $f_\epsilon(1/2) = f_{\epsilon'}(1/2)$ if ϵ is the immediate predecessor of ϵ'. See Figure 3.6.

Let $S(p) = \{f_\epsilon(p) \mid \epsilon \in \{\mathtt{0},\mathtt{1}\}^\omega\}$. By Lemma 3.15, $f_\epsilon(p) \ne f_{\epsilon'}(p)$, if $\epsilon \ne \epsilon'$ and $0 < p < 1/2$. Therefore the mapping $\gamma_p : \{\mathtt{0},\mathtt{1}\}^\omega \to [0,1]$, defined by $\epsilon \mapsto f_\epsilon(p)$ is one-to-one for $0 < p < 1/2$, and thus $S(p) = \gamma_p(\{\mathtt{0},\mathtt{1}\}^\omega)$ is uncountable. However, the mapping is not onto; there are gaps on the image, namely, the open intervals $(\gamma_p(\epsilon), \gamma_p(\epsilon'))$, where

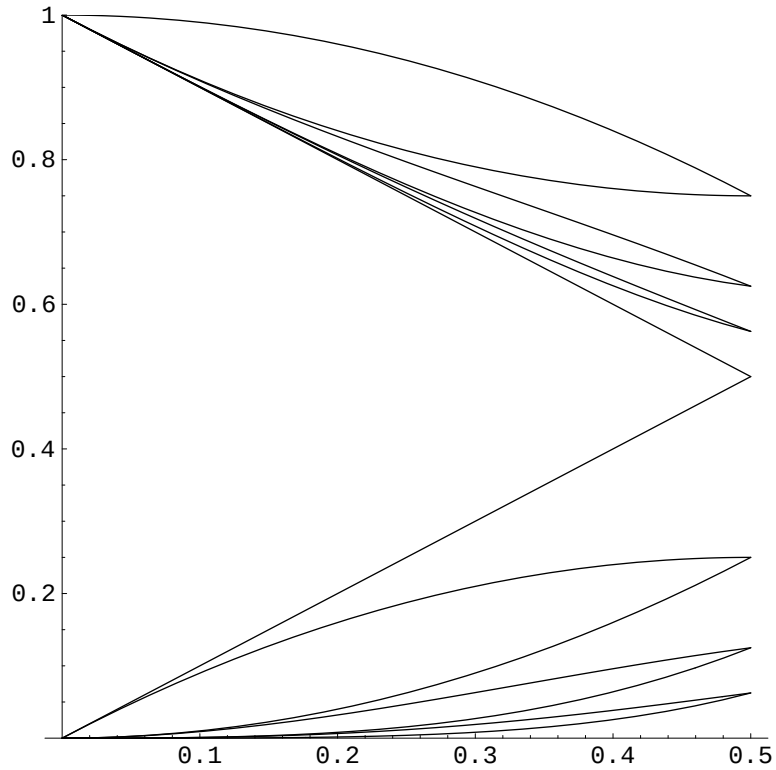


Figure 3.6 Plots of f_ϵ for fourteen sequences: $\mathtt{001}^\omega$, $\mathtt{010}^\omega$, $\mathtt{0101}^\omega$, $\mathtt{0110}^\omega$, $\mathtt{01101}^\omega$, $\mathtt{01110}^\omega$, $\mathtt{01}^\omega$, $\mathtt{10}^\omega$, $\mathtt{101}^\omega$, $\mathtt{110}^\omega$, $\mathtt{1101}^\omega$, $\mathtt{1110}^\omega$, $\mathtt{11101}^\omega$, and $\mathtt{11110}^\omega$, in an increasing order. They correspond to the functions $q{+}qp, q{+}p^2, q{+}qp^2, q{+}p^3, q{+}qp^3, q{+}p^4, q, p, qp, p^2, qp^2, p^3, qp^3$, and p^4, respectively. Note that the pairs of sequences whose corresponding functions are equal at $1/2$ are the pairs of immediate predecessors and successors.

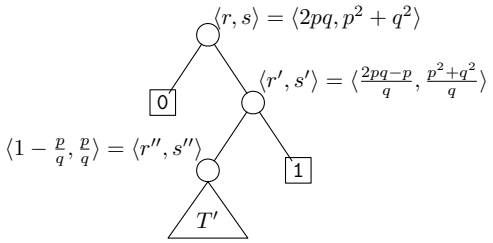
ϵ is the immediate predecessor of ϵ'. These gaps correspond exactly to the middle sets that are cut off in the construction of the Cantor sets. The Hausdorff dimension of $S(p)$ is $-\log 2/\log p$. So we have a class of uncountably many Cantor sets such that for each real number h between 0 and 1, the class contains a Cantor set of Hausdorff dimension h. (See [11] for Hausdorff dimension.)

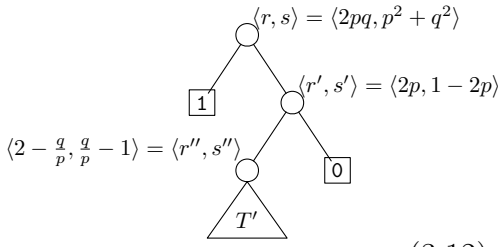
Proposition 3.16. *For $0 < p < 1/2$, $S(p)$ is a Cantor set. The set $S(1/3)$ is the Cantor middle-third set. The Hausdoff dimension of $S(p)$ is $-\log 2/\log p$.*

When $p = 1/2$, γ_p is not one-to-one anymore, but it is onto, and $S(1/2) = [0, 1]$.

3.4.2 Infinite Trees: Set $\mathcal{B}$

Suppose that T is an infinite DDG-tree for the configuration $(p, 2pq)$ and $E(T) = 2$. By Lemma 3.7, (LB3), T takes branching `L` at every internal node from level 1. Since $p < 2pq < q$, both `0` and `1` are possible for the label at level 1. Let T_A and T_B be the DDG-trees with label `0` and `1` at level 1, respectively. Then the target distribution for the sub-DDG-tree rooted at the node at level 1 is $\langle (2pq - p)/q, (p^2 + q^2)/q \rangle$ for T_A, and $\langle 2p, 1 - 2p \rangle$ for T_B. Since $(2pq - p)/q < p$, T_A must take label `1` at the terminal node at level 2. Since $1 - 2p < q$, T_B must take label `0` at the terminal node at level 2. So T is either of the following forms:

$T_A =$ 

$T_B =$ 

(3.12)

where T' is an infinite tree such that every internal node takes branching `L`, whose target bias is $1 - p/q$ in the case of T_A, and $2 - q/p$ in the case of T_B. Note that T' is T_ϵ for some $\epsilon \in \{\mathtt{0}, \mathtt{1}\}^\omega$. The target bias of T_ϵ is $f_\epsilon(p)$. Therefore, for a given p, T_A is a DDG-tree for the configuration $(p, 2pq)$ if and only if $f_\epsilon(p) = 1 - p/q$, and T_B is a DDG-tree for the configuration $(p, 2pq)$ if and only if $f_\epsilon(p) = 2 - q/p$.

Lemma 3.17. *If target distribution of T_ϵ is $\langle 1 - p/q, p/q \rangle$ for some p in $(0, 1/2)$, then $\epsilon_0 = \mathtt{1}$. Moreover, for every ϵ such that $\epsilon_0 = \mathtt{1}$, there is a unique p such that T_ϵ is a DDG-tree for the configuration $(p, 1 - p/q)$.*

Proof. The largest element ϵ in $\{\mathtt{0}, \mathtt{1}\}^\omega$ such that $\epsilon_0 = \mathtt{0}$ is $\mathtt{01}^\omega$. For $0 < p \leq 1/2$, $f_{\mathtt{01}^\omega}(p) = q > 1 - p/q$. Therefore, by Lemma 3.15 for every ϵ in $\{\mathtt{0}, \mathtt{1}\}^\omega$ such that $\epsilon_0 = \mathtt{0}$, $f_\epsilon(p) > f_{\mathtt{01}^\omega}(p) > 1 - p/q$. So there is no $p \in (0, 1/2]$ such that $f_\epsilon(p) = 1 - p/q$. (See

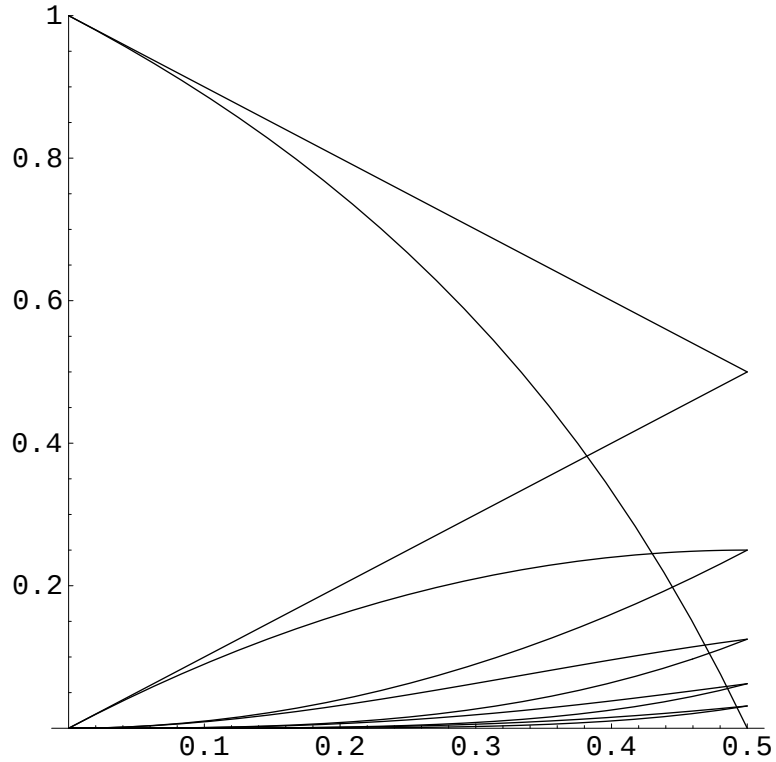


Figure 3.7 Plots of f_ϵ for some $\epsilon \succeq \mathtt{10}^\omega$ and $g(p) = 1 - p/q$.

Figure 3.6.) Now consider ϵ in $\{\mathtt{0},\mathtt{1}\}^\omega$ such that $\epsilon_0 = \mathtt{1}$. Then by Lemma 3.13, $f_\epsilon(0) = 0$ and f_ϵ is monotone increasing. Let us define $g(p) = 1 - p/q$. Note that g is monotone decreasing, $g(0) = 1$ and $g(1/2) = 0$. Hence, the equation $f_\epsilon(p) = g(p)$ has a unique root in $(0, 1/2)$. See Figure 3.7. □

Let $\mathtt{1}\{\mathtt{0},\mathtt{1}\}^\omega = \{\epsilon \in \{\mathtt{0},\mathtt{1}\}^\omega \mid \epsilon_0 = \mathtt{1}\}$, and consider a mapping $\alpha : \mathtt{1}\{\mathtt{0},\mathtt{1}\}^\omega \to (0, 1/2)$ such that $\alpha(\epsilon)$ is the root of the equation $f_\epsilon(p) = g(p)$. That is, $\alpha(\epsilon)$ is the unique source bias p for which T_ϵ is the DDG-tree for the configuration $(p, 1 - p/q)$. The mapping α is strictly increasing with respect to the orders $\prec$ and $<$, and therefore α is one-to-one. Let $\mathcal{B}^A = \alpha(\mathtt{1}\{\mathtt{0},\mathtt{1}\}^\omega)$.

With Lemma 3.17, we established a one-to-one correspondence between $\mathtt{1}\{\mathtt{0},\mathtt{1}\}^\omega$ and the set of DDG-trees of the form T_A of (3.12). A statement similar to Lemma 3.17

holds for the target distribution $\langle 2 - q/p, q/p - 1\rangle$, and in this case there is a one-to-one correspondence between $\{0,1\}^\omega$ and the set of DDG-trees of the form T_B of (3.12). Let $\mathcal{B}^B$ be the set of unique source biases corresponding to each $\epsilon \in \{0,1\}^\omega$. Then $\mathcal{B} = \mathcal{B}^A \cup \mathcal{B}^B$. Since there are uncountably many $\epsilon \in \{0,1\}^\omega$, we conclude the following.

Theorem 3.18. *There are uncountably many infinite DDG-trees for $r = 2pq$, whose average depth is 2. Moreover, each DDG-tree is of the forms shown in (3.12).*

Now we are going to study the structure of the set $\mathcal{B}^A$. For $\mathcal{E}, \mathcal{E}' \subset \{0,1\}^\omega$, let us write $\mathcal{E} \prec \mathcal{E}'$ if $\epsilon \prec \epsilon'$ for each $\epsilon \in \mathcal{E}$ and $\epsilon' \in \mathcal{E}'$. Similarly, for sets of real numbers A and B, let us write $A < B$ if $a < b$ for each $a \in A$ and $b \in B$. Let us define

$$\mathcal{E}_k \;=\; \{\epsilon \,|\, 1^k 0^\omega \preceq \epsilon \prec 1^{k+1}0^\omega\}$$

and let

$$\mathcal{B}^A_k = \alpha(\mathcal{E}_k).$$

for $k = 1, 2, \ldots$. Then we have $1\{0,1\}^\omega \setminus \{1^\omega\} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \cdots$, and $\mathcal{E}_1 \prec \mathcal{E}_2 \prec \cdots$. Because α is strictly increasing $\mathcal{B}^A \setminus \{1/2\} = \mathcal{B}^A_1 \cup \mathcal{B}^A_2 \cup \cdots$, and $\mathcal{B}^A_1 < \mathcal{B}^A_2 < \cdots$.

Now note that each $\mathcal{E}_k$ has a least element and a maximal element, namely, 1^k0^ω and 1^k01^ω, respectively. Let us call them λ_k and μ_k, respectively. Then $\mathcal{E}_k = \{\epsilon \,|\, \lambda_k \preceq \epsilon \preceq \mu_k\}$, $\alpha(\lambda_k) = \min \mathcal{B}^A_k$, and $\alpha(\mu_k) = \max \mathcal{B}^A_k$. Let I_k be the closed interval $[\alpha(\lambda_k), \alpha(\mu_k)]$, for $k = 1, 2, \ldots$. Hence, for each k, $\mathcal{B}^A_k \subset I_k$. Note that

$$\lambda_1 \prec \mu_1 \prec \lambda_2 \prec \cdots \prec \lambda_k \prec \mu_k \prec \lambda_{k+1} \prec \cdots,$$

and, for each k, μ_k is the immediate predecessor of λ_{k+1}. That is, there is no element between μ_k and λ_{k+1}. However, although α preserves the order, $\alpha(\mu_k)$ is not an immediate predecessor of $\alpha(\lambda_{k+1})$; for any two real numbers a, b such that $a < b$, there are uncountably many real numbers!

Let J_0 be the open interval $(0, \alpha(\lambda_1))$, and let $J_k = (\alpha(\mu_k), \alpha(\lambda_{k+1}))$, for $k = 1, 2, \ldots$. Then we have

$$(0, 1/2) = J_0 \cup I_1 \cup J_1 \cup I_2 \cup J_2 \cup \cdots.$$

Note that each J_k is a nonempty open interval and is excluded from $\mathcal{B}^A$. So,

$$\mathcal{B}^A \setminus \{1/2\} \subset (0, 1/2) \setminus (\cup_{k=0}^{\infty} J_k).$$

Now we will show that an infinite number of open subintervals of I_k are not included in $\mathcal{B}^A_k$, for each $k = 1, 2, \ldots$. Let $e_k = \mathtt{1}^{k-1}\mathtt{0}$. Then $\mathcal{E}_k = \{\epsilon \,|\, \mathtt{1}e_k\mathtt{0}^\omega \preceq \epsilon \prec \mathtt{1}e_{k+1}\mathtt{0}^\omega\}$. For each k, we can partition $\mathcal{E}_k$ into an infinite number of disjoint subsets, $\mathcal{E}_{k1}, \mathcal{E}_{k2}, \ldots$, where

$$\mathcal{E}_{kl} = \{\epsilon \,|\, \mathtt{1}e_k e_l \mathtt{0}^\omega \preceq \epsilon \prec \mathtt{1}e_k e_{l+1}\mathtt{0}^\omega\}.$$

More generally, let us define

$$\mathcal{E}_{k_1 \cdots k_m} = \{\epsilon \,|\, \mathtt{1}e_{k_1}e_{k_2}\cdots e_{k_m}\mathtt{0}^\omega \preceq \epsilon \prec \mathtt{1}e_{k_1}e_{k_2}\cdots e_{k_{m-1}}e_{k_m+1}\mathtt{0}^\omega\},$$

for positive integers $k_1, k_2, \ldots, k_m$. Then

$$\mathcal{E}_{k_1 \cdots k_m} = \bigcup_{l=1}^{\infty} \mathcal{E}_{k_1 \cdots k_{m-1} l},$$

and

$$\mathcal{E}_{k_1 \cdots k_{m-1}1} \prec \mathcal{E}_{k_1 \cdots k_{m-1}2} \prec \cdots.$$

The set $\mathcal{E}_{k_1 \cdots k_m}$ has the minimal element $\lambda_{k_1 k_2 \cdots k_m} = \mathtt{1}e_{k_1}\cdots e_{k_m}\mathtt{0}^\omega$ and the maximal element $\omega_{k_1 k_2 \cdots k_m} = \mathtt{1}e_{k_1}\cdots e_{k_m}\mathtt{1}^\omega$. Let $\mathcal{B}^A_{k_1 \cdots k_m} = \alpha(\mathcal{E}_{k_1 \cdots k_m})$, and

$$I_{k_1 k_2 \cdots k_m} = [\alpha(\lambda_{k_1 k_2 \cdots k_m}), \alpha(\omega_{k_1 k_2 \cdots k_m})],\ J_{k_1 k_2 \cdots k_m} = (\alpha(\omega_{k_1 k_2 \cdots k_m}), \alpha(\lambda_{k_1 k_2 \cdots k_m + 1})).$$

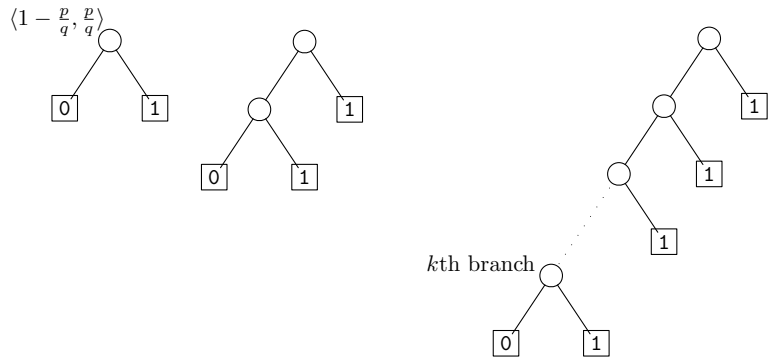
Then

$$\mathcal{B}^A_{k_1 \cdots k_{m-1}} \subset I_{k_1 \cdots k_{m-1}} \setminus \cup_{l=1}^{\infty} J_{k_1 \cdots k_{m-1} l}.$$

For each m, $1\{0,1\}^\omega$ is a disjoint union of the subsets $\mathcal{E}_{k_1\cdots k_m}$, for all possible positive integers $k_1, k_2, \cdots, k_m$. Therefore, given $\epsilon \in 1\{0,1\}^\omega$, there is a unique sequence of positive integers $k_1, k_2, \cdots$ such that ϵ is in $\mathcal{E}_{k_1\cdots k_m}$, for each $m \geq 1$. Suppose that $\epsilon \prec \epsilon'$, and let $(k_1', k_2', \cdots)$ be the corresponding sequence for ϵ'. So there is the smallest l such that $k_l < k_l'$. Hence $\epsilon \in \mathcal{E}_{k_1\cdots k_l}$, and $\mathcal{E}_{k_1\cdots k_l} \prec \epsilon'$, and $\alpha(\epsilon) \in I_{k_1\cdots k_l} < J_{k_1\cdots k_l} < \alpha(\epsilon')$. Therefore we conclude that for any two points in $\mathcal{B}^A$, there is a open interval between them that is not contained in $\mathcal{B}^A$. Hence $\mathcal{B}^A$ is nowhere dense. Similarly, we can show that $\mathcal{B}^B$ is nowhere dense. We have proved part (ii) of Theorem 3.2.

3.4.3 Finite Trees: Sets $\mathcal{C}$ and $\mathcal{D}$

In this section we prove part (iii) of Theorem 3.2. Suppose that T is a finite DDG-tree whose average depth is at most 2, and suppose that T is not T_{2pq}. Since the target bias $r(p)$ of T is not identically equal to $2pq$, and $r(p)$ is a polynomial function of p, there can be only finitely many values of p that satisfy the equation $r(p) = 2pq$. Note that there are only countably many finite DDG-trees. Therefore, we conclude that $\mathcal{C} \cup \mathcal{D}$ is countable. Now let us consider the following family of DDG-trees as candidates for the subtree T' of T_A in (3.12).



(3.13)

Let T_k be the DDG-tree of depth k in (3.13). Then $E(T_k) = 1 + p + p^2 + \cdots + p^{k-1} = (1-p^k)/q < 1/q$. Therefore, $E(T_A) = 1 + qE(T')$ is strictly less than 2, when $T' = T_k$, for $k = 1, 2, \ldots$.

The target bias of T_k is p^k, for $k = 1, 2, \ldots$. Therefore, for a given p, the DDG-tree T_A with $T' = T_k$ is a DDG-tree for the configuration $(p, 2pq)$ if and only if

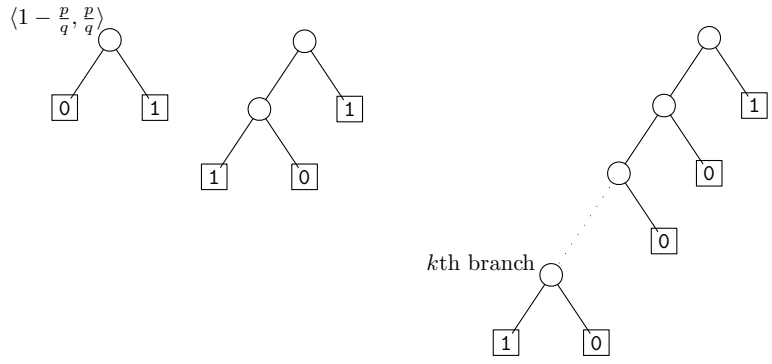
$$1 - p/q = p^k. \tag{3.14}$$

As in the proof of Lemma 3.17, there is a unique root for equation (3.14) for each k, for $0 < p \leq 1/2$. Moreover, for each k, the roots are distinct. (Actually, the root is $\alpha(\mu_k)$ since $f_{\mu_k}(p) = p^k$. Note that α is one-to-one. The roots for first several k's correspond to the small circles on the curve in Figure 3.5.) Therefore, there are infinitely many values of p for which there is a finite DDG-tree for the configuration $(p, 2pq)$ whose average depth less is than 2. So we completed the proof of part (iii) of Theorem 3.2.

3.5 Remarks

3.5.1 Shape of C_{2pq}

The proof of Theorem 3.3 relies on the topological property of $\tilde{F^*}^{-1}(\mathtt{B})$ near $p = 1/2$, or the fact that there are infinitely many holes in C_{2pq}. In fact, the shape of C_{2pq} is much more complicated than shown in Figure 3.5. Consider the following family of trees:



(3.15)

When each tree in (3.15) replaces T' of T_A in (3.12), the average depths of the resulting trees are less than 2, and the corresponding values of p have an accumulation point near 0.382. Moreover, there are infinitely many such families of trees.

3.5.2 Model of Recursive Construction of DDG-tree

We considered the DDG-branching function based on seven bases cases of branching. Instead of seven cases, we can use a model of recursive construction of DDG-trees based on three base cases of branching: stop with output 0 or 1, or branch. We will show that two models have the same expressive power. The same impossibility result holds for his model using essentially the same proof. I will refer the two models as the *new* model and the *old* model.

Let G be a DDG-branching function based on the new model, and it takes a configuration (source bias and target bias) (p, r) as input and outputs $\boxed{0}, \boxed{1}$ or a tuple (r_1, r_2), where r_1 and r_2 are target biases of the subtrees. Using this function, a DDG-tree can be constructed in a similar manner as with the old model. For example, Knuth-Yao construction can be expressed by the following function:

$$G_{KY}(1/2, r) = \begin{cases} \boxed{1} & \text{if } r = 0 \\ \boxed{0} & \text{if } r = 1 \\ (0, 1) & \text{if } r = \frac{1}{2} \\ (2r, 0) & \text{if } r < \frac{1}{2} \\ (2r - 1, 1) & \text{if } r > \frac{1}{2} \end{cases} \tag{3.16}$$

Compare this with the DDG-branching function F_{KY} for Knuth-Yao method with respect to the old model on page 43.

$$F_{KY}(1/2, r) = \begin{cases} (\texttt{S0}, -) & \text{if } r = \frac{1}{2} \\ (\texttt{L0}, 2r) & \text{if } r < \frac{1}{2} \\ (\texttt{L1}, 2r - 1) & \text{if } r > \frac{1}{2} \end{cases} \tag{3.17}$$

Note that a DDG-branching function G with respect to the new model must satisfy the condition that $G(p, 0) = \boxed{1}$ and $G(p, 1) = \boxed{0}$. Otherwise, G would not induce a legitimate DDG-tree. Then we have a one-to-one correspondence between DDG-branching

functions based on the new model and DDG-branching functions based on the old model by substituting the values of the functions by the following correspondence:

$$\begin{aligned}
(0,1) &\leftrightarrow (\texttt{S0},-)\\
(1,0) &\leftrightarrow (\texttt{S1},-)\\
(r_1,0) &\leftrightarrow (\texttt{L0},r_1)\\
(r_1,1) &\leftrightarrow (\texttt{L1},r_1)\\
(0,r_2) &\leftrightarrow (\texttt{R0},r_2)\\
(1,r_2) &\leftrightarrow (\texttt{R1},r_2)\\
(r_1,r_2) &\leftrightarrow (\texttt{B},r_1).
\end{aligned}$$

The branch part of the functions (such as "if $r < \frac{1}{2}$") does not change in this correspondence.

So the expressive powers of the two models are identical. Although the seven cases of the old model seem to be more complicated, it is just a different way of expressing the branchings with respect to the new (simpler-looking) model. In other words, the DDG-branching function using the new model needs to specify both the target biases r_1 and r_2 of the subtrees instead of not specifying `L` and `R`. Since the expressive powers are identical, if recursive construction using algebraic decisions (the "if" part of the DDG-branching functions) of optimal DDG-tree is impossible with respect to the old model, then it is impossible with respect to the new model, and vice versa.

CHAPTER 4

Entropy Bound

In this chapter we give an elementary proof of the entropy bound for the average depth of DDG-tree algorithms. Consider an n-ary DDG-tree for an m-ary target distribution. The DDG-tree induces a function $f : \mathcal{T} \to \{0, \ldots, m-1\}$, where $\mathcal{T}$ is an exhaustive prefix-free set over $\{0, \ldots, n-1\}$ corresponding to the DDG-tree. In this chapter, we call such a function a *tree function*. We call a tree function is for the configuration $(\mathbf{p}, \mathbf{r})$ if the corresponding DDG-tree simulates the target distribution $\mathbf{r} = \langle r_1, \ldots, r_m \rangle$ using the source distribution $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$. The average depth of a tree function is the average depth of the corresponding DDG-tree.

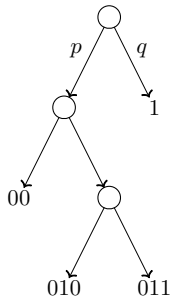
Theorem 4.1. *For the source distribution $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$ and the target distribution $\mathbf{r} = \langle r_1, \ldots, r_m \rangle$, the average depth of a tree function for $(\mathbf{p}, \mathbf{r})$ is at least $\mathcal{H}(\mathbf{r})/\mathcal{H}(\mathbf{p})$, where $\mathcal{H}$ is the entropy function.*

Several versions of this theorem appear in literature: Elias [10] and Peres [28] for the generation of a fair coin from a biased coin, Knuth and Yao [20] and Cover and Thomas [7, Section 5.12] for the generation of a general discrete distribution from a fair coin, and Roche [29] and Han and Hoshi [13] for the generation of a general discrete distribution from another general discrete distribution, as in our case.

We can see Theorem 4.1 as a corollary of Theorem 4.2 below. The main purpose of this chapter is to give an elementary proof of Theorem 4.2. The theorem is known as "leaf entropy" theorem in a more general form [22]. Cover and Thomas [7, Theorem 5.12.1] prove a special case of Theorem 4.2, in which the source distribution is two-valued and uniform. Our proof is interesting because it is purely algebraic.

Entropy of an Induced Random Variable Let X be a random variable that takes values over $\Sigma = \{x_1, \ldots, x_n\}$ such that $\Pr[X = x_i] = p_i$ for each $i = 1, \ldots, n$. Consider a random variable Y over an exhaustive prefix-free subset $\mathcal{T}$ of Σ^* such that for $\sigma = x_{i_1} \cdots x_{i_k}$ in $\mathcal{T}$, $\Pr[Y = \sigma] = p_{i_1} \cdots p_{i_k}$. Since $\mathcal{T}$ is exhaustive, $\sum_{\sigma \in \mathcal{T}} \Pr[Y = \sigma] = 1$. We say that Y is *induced* from X via $\mathcal{T}$.

An induced random variable can be represented as a complete n-ary tree. Conversely, a complete n-ary tree defines an induced random variable. As an example, let $\Sigma = \{0, 1\}$ with probabilities $\langle p, q \rangle$. The following tree represents an induced random variable over $\{1, 00, 010, 011\}$, whose probability distribution is $\langle q, p^2, p^2q, pq^2 \rangle$.



Now let $D = |Y|$. Then D is the random variable representing the length of a word in $\mathcal{T}$. So $E(D)$, the expected value of D, is the average length of the words in $\mathcal{T}$, or equivalently, the average depth of the tree corresponding to the induced random variable Y. We will show that the entropy of the induced random variable Y equals the entropy of the random variable X multiplied by $E(D)$.

Theorem 4.2. *Let Y be a random variable induced from X and let $D = |Y|$. Then $\mathcal{H}(Y) = E(D)\mathcal{H}(X)$.*

Proof. We first present a proof in the case $n = 2$. Because the probabilities for the words in $\mathcal{T}$ sum to 1, with a slight abuse of notation we can write

$$1 = \sum_{\mathcal{T}} p^k q^l,$$

where k is the number of left edges taken in the path from the root to a terminal node of the tree (or the number of 0's in the corresponding word in $\mathcal{T}$), and l is the number right edges, hence the probability $p^k q^l$ for a terminal node. The average length of the words in $\mathcal{T}$ is

$$E(D) = \sum_{\mathcal{T}} (k + l) p^k q^l.$$

Consequently,

$$-E(D)\mathcal{H}(X) = \sum (k + l) p^k q^l p \log p + \sum (k + l) p^k q^l q \log q.$$

Now the entropy of Y is

$$\begin{aligned} \mathcal{H}(Y) &= -\sum p^k q^l \log(p^k q^l) \\ &= -\sum k p^k q^l \log p - \sum l p^k q^l \log q \end{aligned}$$

To prove Theorem 4.2 for $n = 2$, it suffices to prove the following equalities.

$$p \sum (k + l) p^k q^l = \sum k p^k q^l \tag{4.1}$$

$$q \sum (k + l) p^k q^l = \sum l p^k q^l \tag{4.2}$$

We will prove the equality (4.1). Consider the function

$$F(x) = \sum_{\mathcal{T}} x^k (1 - x)^l.$$

The function $F(x)$ is a polynomial in the case $\mathcal{T}$ is finite, and a power series when $\mathcal{T}$ is infinite. By definition, F is identically 1 on $[0,1]$. (Actually, it does not have to be restricted on the interval $[0,1]$.) Therefore, the first derivative of F is identically zero:

$$F'(x) = \sum_{\mathcal{T}} \left[kx^{k-1}(1-x)^l - lx^k(1-x)^{l-1} \right] \equiv 0.$$

Hence for every p in $[0,1]$, we obtain an identity

$$\sum_{\mathcal{T}} kp^{k-1}q^l = \sum_{\mathcal{T}} lp^k q^{l-1}. \tag{4.3}$$

The identity (4.3) is used in the following manipulation of equations, which proves the identity (4.1).

$$\begin{aligned} p\sum(k+l)p^k q^l &= pq\sum kp^k q^{l-1} + pq\sum lp^k q^{l-1} \\ &= pq\sum kp^k q^{l-1} + pq\sum kp^{k-1}q^l \\ &= \sum kp^k q^l(p+q) \\ &= \sum kp^k q^l \end{aligned}$$

The equation (4.2) is proved similarly.

For a general n-valued distribution $\langle p_1, \ldots, p_n \rangle$, we consider the function of the form

$$F(x_1, \ldots, x_n) = \sum_{\mathcal{T}} x_1^{k_1} \cdots x_n^{k_n},$$

which is identically 1 on the hyperplane defined by the equation $x_1 + \cdots + x_n = 1$. Then by taking a partial derivative at a point $(p_1, \ldots, p_n)$ on the same hyperplane, we have

$$\begin{aligned} \frac{\partial F}{\partial x_i}(p_1, \ldots, p_n) &= \sum k_i p_1^{k_1} \cdots p_i^{k_i - 1} \cdots p_n^{k_n} \\ &\quad - \sum k_n p_1^{k_1} \cdots p_n^{k_n - 1} = 0. \end{aligned}$$

As a result, we obtain the following identity:

$$\begin{aligned} \sum k_1 p_1^{k_1 - 1} \cdots p_n^{k_n} &= \cdots = \sum k_i p_1^{k_1} \cdots p_i^{k_i - 1} \cdots p_n^{k_n} \\ &= \cdots = \sum k_n p_1^{k_1} \cdots p_n^{k_n - 1}. \end{aligned}$$

With this identity, the proof follows for an n-valued distribution as in the 2-valued case. □

Proof of Theorem 4.1 Theorem 4.1 follows from Theorem 4.2. Let X be a random variable with alphabet $\Sigma = \{x_1, \ldots, x_n\}$ with distribution $\mathbf{p}$. If $f : \mathcal{T} \to \{y_1, \ldots, y_m\}$ is a tree function for $(\mathbf{p}, \mathbf{r})$, then $f(Y)$ is a random variable over $\{y_1, \ldots, y_m\}$ with probability distribution $\mathbf{r}$, where Y is induced from X via $\mathcal{T}$. Note that the average cost of a tree function is the average depth of the corresponding tree. Since $\mathcal{H}(f(Y)) \leq \mathcal{H}(Y)$, we have proved Theorem 4.1.

CHAPTER 5

Conclusions and Future Work

5.1 Conclusions

We defined randomizing functions to study the simulation of a target discrete probability distribution using a source of unknown distribution. A randomizing function is immune to the source distribution. A key observation is that a randomizing function should exploit the fact that certain inputs are equally likely (Theorem 2.9 and Theorem 2.13). We proved that among randomizing functions that take source inputs of a fixed length and simulate a uniform target distribution, Elias's functions are optimal. We generalized Elias's functions to obtain optimal randomizing functions that simulate arbitrary rational target distributions.

We studied randomizing functions on exhaustive prefix-free sets and observed that the generalizations of Elias's functions are not always optimal. We stated a condition on exhaustive prefix-free sets for which Elias's functions are optimal.

We showed that the optimal randomizing functions are efficiently computable.

On the other hand, in the case of known source distribution, we established a difficulty of optimal simulation. We considered a recursive construction based on algebraic computations that deals with the simplest instance of the problem: the simulation of

a coin of bias r using a coin of bias p. The model is almost the only sensible way of computing the simulation because the simulation tree is infinite in most cases, and the previously known methods can be carried out with this model. We used a topological argument to show that it is impossible to obtain an optimal simulation using the recursive construction based on classical algebraic computations.

5.2 Future Work and Open Questions

Randomizing Functions and Universal Coding The idea of a randomizing function is closely related to the idea of *universal source coding*, an encoding scheme independent of the source distribution. In fact, the procedure we considered in this paper can be modified to obtain a universal source coding. In the binary case, for example, consider the mapping $x \mapsto (k, r(x))$, where k is the number of T's in x and $r(x)$ is the rank of x in S_k. Clearly, this mapping gives a uniquely decodable code, and we have seen that it is efficiently computable. Also the inverse mapping (decoding) is efficiently computable. It is not hard to show that the average code length approaches $\mathcal{H}(p) = -p \log_2 p - q \log_2 q$ as the block size n increases regardless of the parameter p. So this code is asymptotically optimal. It is not accidental that the computation of ranking we considered in Section 2.4 was considered in the context of source coding [6, 21, 30].

Impossibility and the Computation Model Blum et al. [3] consider a more powerful model of real number computation that allows loops as well as primitive algebraic operations and tests. If a function is computable with their model, then the inverse image of a finite set is a countable union of semi-algebraic sets. One open question is whether an optimal DDG-branching function can be computed with the Blum-Shub-Smale model. The key argument in the proof of Theorem 3.3 is that the set C has an

infinite number of connected components. If we can show, for example, that the set C has an uncountable number of connected components, then we can conclude that an optimal DDG-branching function cannot be computed with the Blum-Shub-Smale model, which is a stronger result than Theorem 3.3.

Although we do not know the answer to the question, we know that the set $\mathcal{B}$ has uncountably many connected components (in fact, $\mathcal{B}^A$ and $\mathcal{B}^B$ in Section 3.4 are homeomorphic to the Cantor ternary set), and therefore it is undecidable with respect to the Blum-Shub-Smale model. Note that the average depth an infinite tree for configuration $(p, 2pq)$ is at least 2 (Lemma 3.11). So the set $\mathcal{B}$ corresponds exactly to optimal infinite DDG-trees for configuration $(p, 2pq)$. As a consequence, for example, the following question is undecidable with respect to this model:

> *Given a configuration (p, r), is there an optimal infinite DDG-tree for the configuration?*

As another question regarding our model of computation, we could ask what if a DDG-tree is allowed to output more than one bit. (In Chapter 2, we considered functions that output more than one bit.) We conjecture that we can obtain a similar impossibility result.

REFERENCES

[1] J. Abrahams. Generation of discrete distributions from biased coins. *IEEE Transactions on Information Theory*, 42(5):1541–1546, 1996.

[2] M. Ben-Or. Lower bounds for algebraic computation trees (preliminary report). In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 80–86, Boston, Massachusetts, 25–27 Apr. 1983.

[3] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer Verlag, 1998.

[4] M. Blum. Independent unbiased coin flips from a correlated biased source: A finite state Markov chain. *Combinatorica*, 6(2):97–108, 1986.

[5] B. A. Cipra. Tossed coins and troubled marriages: Mathematical highlights from AAAS 2004. *SIAM News*, 37(4), May 2004.

[6] T. M. Cover. Enumerative source encoding. *IEEE Transactions on Information Theory*, 19(1):73–77, January 1973.

[7] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA, 1991.

[8] P. Diaconis. The search for randomness. at American Association for the Advancement of Science annual meeting. Feb. 14, 2004. Seattle.

[9] E. W. Dijkstra. Making a fair roulette from a possibly biased coin. *Information Processing Letters*, 36:193, 1990.

[10] P. Elias. The efficient construction of an unbiased random sequence. *The Annals of Mathematical Statistics*, 43(3):865–870, 1972.

[11] K. J. Falconer. *The Geometry of Fractal Sets*. Cambridge University Press, 1985.

[12] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Springer, 1999.

[13] T. S. Han and M. Hoshi. Interval algorithm for random number generation. *IEEE Transactions on Information Theory*, 43(2):599–611, 1997.

[14] W. Hoeffding and G. Simons. Unbiased coin tossing with a biased coin. *The Annals of Mathematical Statistics*, 41:341–352, 1970.

[15] A. Juels, M. Jakobsson, E. Shriver, and B. K. Hillyer. How to turn loaded dice into fair coins. *IEEE Transactions on Information Theory*, 46(3):911–921, 2000.

[16] B. Jun and P. Kocher. The Intel random number generator. White paper prepared for Intel Corporation, 1999.

[17] E. Klarreich. Toss out the toss-up: Bias in heads-or-tails. *Science News*, 165:131–132, 2004. Available at http://www.sciencenews.org/articles/20040228/fob2.asp.

[18] D. Knuth. *The Art of Computer Programming*, volume 4A. Addison-Wesley, 2004. Pre-fascicle 3a, Generating all combinations, available at http://www-cs-faculty.stanford.edu/~knuth/taocp.html.

[19] D. E. Knuth. *The Art of Computer Programming, Seminumerical Algorithms*, volume 2. Addison-Wesley, third edition, 1998.

[20] D. E. Knuth and A. C.-C. Yao. The complexity of nonuniform random number generation. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results. Proceedings of a Symposium*, pages 357–428, New York, NY, 1976. Carnegie-Mellon University, Computer Science Department, Academic Press. Reprinted in Knuth's *Selected Papers on Analysis of Algorithms* (CSLI, 2000).

[21] J. C. Lawrence. A new universal coding scheme for the binary memoryless source. *IEEE Transactions on Information Theory*, 23(4):466–472, July 1977.

[22] J. L. Massey. The entropy of a rooted tree with probabilities. In *Proceedings of the 1983 IEEE International Symposium on Information Theory*, 1983.

[23] J. Milnor. On the Betti numbers of real varieties. *Proceedings of the American Mathematical Society*, 15:275–280, 1964.

[24] N. Nisan and A. Wigderson. Hardness vs Randomness. *J. Comput. System Sci.*, 49:149–167, 1994.

[25] O. A. Oleinik and I. B. Petrovskii. On the topology of real algebraic surfaces. *Izv. Akad. Nauk SSSR*, 13:389–402, 1949.

[26] S. Pae. The optimality of random number generation from a biased coin. 2004. submitted.

[27] S. Pae and M. C. Loui. Optimal random number generation from a biased coin. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1079–1088, January 2005.

[28] Y. Peres. Iterating von Neumann's procedure for extracting random bits. *Annals of Statistics*, 20(1):590–597, 1992.

[29] J. R. Roche. Efficient generation of random variables from biased coins. Technical report, AT&T Lab., 1992. Bell Tech. Report File case 20878.

[30] J. P. M. Schalkwijk. An algorithm for source coding. *IEEE Transactions on Information Theory*, 18(3):395–399, May 1972.

[31] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. The University of Illinois Press, Urbana, 1964.

[32] J. M. Steele and A. C. Yao. Lower bounds for algebraic decision trees. *Journal of Algorithms*, 3(1):1–8, Mar. 1982.

[33] Q. F. Stout and B. Warren. Tree algorithms for unbiased coin tossing with a biased coin. *Annals of Probability*, 12(1):212–222, 1984.

[34] R. Thom. Sur l'homologie des variétés algébriques réelles. In S. S. Cairns, editor, *Differential and Combinatorial Topology*, pages 255–265. Princeton University Press, 1965.

[35] J. von Neumann. Various techniques for use in connection with random digits. Notes by G. E. Forsythe. In *Monte Carlo Method, Applied Mathematics Series*, volume 12, pages 36–38. U.S. National Bureau of Standrads, Washington D.C., 1951. Reprinted in von Neumann's *Collected Works* **5** (Pergammon Press, 1963), 768–770.

[36] A. C.-C. Yao. Decision tree complexity and Betti numbers. *Journal of Computer and System Sciences*, 55(1):36–43, Aug. 1997.

VITA

Sung-il Pae was born in Jinju, Korea on October 6, 1967. He attended Seoul National University and graduated with B.S. in Mathematics in 1993, and he completed his mandatory service in Korean air force in the meanwhile. He received a Master's degree in Mathematics from University of Illinois at Urbana-Champaign in 1997. He continued his study in Computer Science at University of Illinois at Urbana-Champaign and received his Ph.D.