Homophonic Coding and Random Number Generation

Valdemar C. da Rocha Jr., Danielle P. B. de A. Camara, and Cecilio Pimentel

***Abstract*—Efficient generation of a discrete probability distribution is of current interest in areas like cryptography and random number generation. This paper revisits some known homophonic coding techniques and discusses their application in random number generation. Both standard and constrained homophonic coding techniques are considered. Three algorithms are given for generating a discrete probability distribution using one or more biased coins. This approach contributes an alternative solution to the classical problem of generating a discrete probability distribution using biased coins.**

***Index Terms*—Constrained homophonic coding, cryptography, discrete probability distribution, entropy, random number generation, recursive tree algorithms.**

I. Introduction

The homophonic coding technique when applied to a sequence of symbols $(u_1, u_2, \ldots)$, at the output of an information source, produces at the homophonic encoder output a sequence of symbols called homophones in a larger alphabet, i.e., more than one homophone can be one-to-one associated to a given source symbol. Each homophone is usually represented one-to-one by a block $\mathbf{x}_i$, called homophonic codeword, containing W_i D-ary symbols, where D is a positive integer greater than or equal to 2.

Homophonic coding is employed with the objective of decomposing each source symbol probability in such a way that the resulting sequence of homophones (or labels) appear to be randomly generated, i.e., a sequence of independent and identically distributed (i.i.d.) random variables. For simplicity and practical interest this paper focus on binary systems although the procedures described are applicable to D-ary coding alphabets. In a *perfect* standard binary homophonic coding scheme the symbols in each homophonic codeword are i.i.d. binary random variables while in a binary-constrained homophonic coding scheme they are independent and identically distributed but not equally likely binary random variables.

The generation of a string of random variables drawn from a discrete probability distribution using flips of a biased coin is considered an old problem of great importance in the areas of cryptography and random number generation. Random numbers find many applications in practice, in particular they are used to perform tests and simulation of communication systems as well as many other computational applications [1]. Von Neumann [2] introduced a simple algorithm to generate a string of i.i.d. bits from flips of a coin with unknown bias. Since then several researchers have considered and studied the generation of uniform random variables under a variety of different assumptions [3]- [13].

This paper revisits some known homophonic coding techniques and discusses their application in random number generation. Two binary-constrained algorithms are discussed for the representation of discrete probability distributions by tossing a single biased coin. For a given discrete probability distribution $\{p_1, p_2, \ldots, p_K\}$ each entry p_i, $1 \leq i \leq K$, is represented by a sum (finite or infinite) of fractions, where each fraction is labeled by a sequence of i.i.d. random variables. Each fraction is a rational number b_i/n_i where b_i is a number between 1 and D, and n_i is a power of D. A third algorithm, the Λ-algorithm, is presented to generate a sequence of random variables drawn from a discrete probability distribution by tossing two or more distinct biased coins.

The optimality question for the algorithms considered here remains open [11], [14]. However, it is shown by some simple examples that with the Λ-algorithm it is possible to obtain equivalent and in some cases even better results than those in [15]. It is important to notice that the biased coins employed in the Λ-algorithm are arbitrary, i.e., they do not depend on probability distribution $\{p_1, p_2, \ldots, p_K\}$ as is the case with the algorithm in [15]. This approach contributes an alternative solution to the classical problem of generating a sequence of i.i.d. random variables using two or more coins where some of the coins are biased.

We call the reader's attention to the fact that, when the probability distribution of the source of random numbers is known, it was shown that it is *impossible* to construct a recursive tree algorithm to achieve a specific target probability distribution.

In *Section II* we describe the *Maximum entropy per step* algorithm [16], or MAX-ENT algorithm for short, in the context of generation of a discrete probability distribution using a biased coin. In *Section III* we describe the *minimum entropy per step* algorithm [17], also in the context of random number generation, which is later modified and employed in *Section IV* for the generation of a discrete probability distribution using two or more biased coins. In this section, the application of our proposed algorithm will be illustrated through some examples, in which a uniform probability distribution is generated using two coins, one fair coin and a biased coin. Summing up, in *Section V* we will present some conclusions as well as some suggestions for future research.

A. Related Works

Feldman et al. [18] proved, among several results, that the outcomes of an n-sided fair die, that is, the outcome of a

The authors are with the Communications Research Group, Department of Electronics and Systems, Federal University of Pernambuco, 50740-550, Recife, PE, BRAZIL, vcr@ufpe.br, dpbac@ieee.org, cecilio@ufpe.br.

Digital Object Identifier 10.14209/jcis.2015.1

random variable which takes n equiprobable values, can be simulated in bounded time by using flips of just one type of coin of appropriate rational bias if and only if n is a power of 2 ([18], Theorem 2). It also followed from [18] that a general n-sided fair die can always be simulated by using two coins of the appropriate bias and at most $\lceil 2 \log n \rceil + 1$ coin flips, where $\lceil x \rceil$ denotes the smallest integer number greater or equal to x. Gargano and Vaccaro [15] published an improvement on this bound, and considered several algorithmic questions, related to the classical problem of simulating the outcomes of a uniform random variable by using a small number of biased coins, and an algorithm was given to generate an *n*-sided fair die using only a fair coin and a biased coin.

In [16] the algorithm later called MAX-ENT algorithm was introduced to perform homophonic coding in which the symbols in each homophonic codeword are independent and identically distributed binary random variables obeying an arbitrary probability distribution $\Pi_2 = \{p, 1-p\}$, where $p \geq 1/2$. The MAX-ENT algorithm provides a solution to the problem posed by Knuth and Yao [11, page 427] on the generation of probability distributions using a biased coin and advanced one step further the solution originally proposed by Julia Abrahams [3]. In [17] the *minimum entropy per step* algorithm was introduced to perform homophonic coding, having as its main motivation the fact that in many situations it was more efficient than the MAX-ENT algorithm.

In [19] the authors provided the first known algorithm that generates unbiased random bits from an arbitrary finite Markov chain, which operates in expected linear time and achieves the information-theoretic upper bound on efficiency. In [20] the authors considered the problem of generating random bits from a loaded die as a natural generalization of generating random bits from a biased coin, and in this manner enabling the application of existing algorithms to general sources. Furthermore, the authors also investigated new approaches for efficiently generating a prescribed number of random bits from an arbitrary biased coin. In [21] the problem of extracting a prescribed number of random bits was addressed by reading the smallest possible number of symbols from a source whose statistical behaviour is not fully specified. The related interval algorithm proposed by Han and Hoshi [9] has asymptotically optimal performance, however it assumes that the distribution of the input stochastic process is known. It was noticed that, in practice, sources of randomness have inherent correlations and are affected by measurement's noise. In other words, it is difficult to obtain an accurate estimate of the probability distribution. The authors main contribution is the design of extractors that have a variable input-length and a fixed output length, which are efficient in the consumption of symbols from the source, and are capable of generating random bits from general stochastic processes and approach the information theoretic upper bound on efficiency. In [22], the authors' main contribution is an algorithm that generates random bit streams from biased coins, uses bounded space and runs in expected linear time. The algorithm approaches the information theoretic upper bound on efficiency as the size of the allotted space increases. Finally, in [23] the author reports a computation of the exact output rate of a recently discovered generalization of Peres algorithm [24] for generating random bits from loaded dice. Instead of resorting to a brute-force computation for all possible inputs, which becomes quickly impractical as the input size increases, the author computes the total output length on equiprobable sets of inputs by dynamic programming using a recursive formula.

II. The Maximum-Entropy per Step Algorithm

In [11, page 427] a question was posed relating the generation of discrete probability distributions from a biased coin: "What if the source of independent random bits is biased towards 1 with probability p ?" This question was answered in part by Julia Abrahams [3], for the case where the biased binary random variable has the special form (t^i, t^j), for integers i and j, and where t is a positive root of the equation $t^i + t^j = 1$.

We consider in the sequel discrete probability distributions $Q = \{q_1, q_2, \ \ldots, q_K\}$. We assume with no loss of essential generality that all K probabilities $q_i, \ 1 \leq i \leq K$, have non-zero values and that $K \geq 2$. The set of labels $V_i = \{v(i,1), v(i,2), \ldots, v(i,j), \ldots\}$, either finite or countably infinite, associated with $q_i, \ 1 \leq i \leq K$, is characterized by the fact that for each entry $v(i,j)$ we have $P_{V_i|Q}(v(i,j)|q_l) \neq 0$ if and only if $l = i$. For binary variable-length coding of each $v(i,j)$ a sequence $X(1,j), X(2,j), \ldots, X(W_j,j)$ is defined whose entries are binary random variables, taking value in the alphabet $\{0,1\}$, and where W_j, the length of the sequence representing $v(i,j)$, is in general also a random variable. It is required that $x(1,j), x(2,j), \ldots, x(W_j,j)$ be a prefix-free encoding of $v(i,j)$, i.e., such sequences are all distinct and none is the prefix of another. Hereafter all entropies are assumed to be in bits and all logarithms are understood to be in base 2. In order to simplify the notation, we will represent $X(1,j), X(2,j), \ldots, X(W_j,j)$ by $X_1 X_2 \ldots X_W$ whenever no ambiguities result.

Definition 1: We define a biased coin tossing coding scheme to be perfect if the symbols of any sequence $X_1 X_2 \ldots X_W$ are i.i.d. discrete random variables.

Definition 2: We define a biased coin tossing coding scheme to be optimum if it is both perfect and minimizes the average sequence length $E(W)$ over perfect biased coin tossing schemes, for a given discrete probability distribution.

A. Biased Coin Tossing

In standard unbiased D-ary coin tossing schemes, $D \geq 2$, the designer benefits from the fact that a given probability $q_i \in Q, \ 0 < q_i < 1$, has an essentially unique base D decomposition. This follows because q_i either has a unique decomposition as an infinite sum of negative powers of D, or it has both a decomposition as a finite sum of distinct negative powers of D and a decomposition as an infinite sum of distinct negative powers of D in which the smallest term in the finite decomposition is expanded as an infinite sum of successive negative powers of D. For example, for $D = 3$, $q_i = 4/9$ can be decomposed as either $q_i = 1/3 + 1/9$ or as $q_i = 1/3 + (1/27)\sum_{i=0}^{\infty}(2/3)^i$.

Biased coin tossing schemes unfortunately do not inherit the essentially unique probability decomposition property described earlier. This means that in order to split each probability in a discrete probability distribution into labels we need to work with the set of probabilities for all symbols $i,\ 1 \leq i \leq K$, instead of working with only one symbol probability at a time as for the D-ary case. We handle this situation with the biased coin coding algorithm introduced in [16] in the context of homophonic coding and described next in terms of random number generation.

B. Biased Coin Tossing Algorithm

Let $\Pi_2 = \{p, 1-p\},\ p \geq 1/2$, be the biased coin probability distribution. For a given discrete probability distribution the biased coin MAX-ENT algorithm simultaneously finds the decomposition of each probability as a sum (finite or infinite) of terms $p^{\lambda}(1-p)^{l-\lambda}$, and the corresponding prefix-free sequence, where λ is the number of heads (1's) and $l-\lambda$ is the number of tails (0's) of a sequence of length l. The labels $v(i,j)$ are selected as terminal nodes in T, the binary rooted tree with nodes labeled by probabilities, such that from any non-terminal node two branches emanate with probabilities p and $1-p=\overline{p}$, respectively. Let $\alpha(i,j)$ denote the probability of $v(i,j)$.

Definition 3: We define the *component running sum* $\gamma_m(i,j)$ for $q_i,\ 1 \leq i \leq K$, at the m^{th} iteration of the MAX-ENT algorithm as

$$\gamma_m(i,j) = q_i - \sum_{k=1}^{j} \alpha(i,k),$$

with $\gamma_m(i,j) = q_i$ for $j=0$, where j denotes the number of labels allocated to q_i up to the m^{th} iteration.

Definition 4: We define the *running sum set* Γ_m at the m^{th} iteration of the MAX-ENT algorithm as

$$\Gamma_m = \{\gamma_m(i,j) | \gamma_m(i,j) > 0, 1 \leq i \leq K\},$$

with $\Gamma_0 = \{q_1, q_2, \ldots, q_K\}$.

Let $\gamma_{\max} = \max\ \gamma_m(i,j) \in \Gamma_m,\ 1 \leq i \leq K$. We expand each unused (not yet labeled) terminal node in T, whose probability exceeds $\gamma_{\max}$, by the least number of branches sufficient to make the resulting extended terminal node probability less than or equal to $\gamma_{\max}$. We call the resulting tree the *processed binary rooted tree with probabilities* and denote it as T_p. At each iteration labels are assigned to terminal nodes of the corresponding processed binary rooted tree with probabilities, in a manner that the unused terminal node with largest probability is assigned as a label to the symbol with largest running sum $\gamma_{\max}$. The MAX-ENT algorithm consists of the following steps.

1) Let $m=1$. Let Γ_1 be the set whose elements are the probabilities $q_i,\ 1 \leq i \leq K$, ordered in decreasing order, and construct the corresponding processed binary rooted tree with probabilities T_p.
2) Without loss of essential generality, assume that $q_r = \gamma_{\max} \in \Gamma_1$. If there are two or more probabilities with the same largest value, just pick any one of them at random to start. Let $(i,j)=(i,1)$ and let $\gamma_1(i,1) = q_i,\ 1 \leq i \leq K$.
3) Find the unused path E_l of length l in T_p whose probability $P(l)$ is largest among unused paths.
4) Associate to q_r the label (terminal node) $v(r,j)$ and the binary sequence of length $l_{r,j}=l$, whose digits constitute the labeling of E_l in T_p. This implies $\alpha(m,j) = P(l)$. Let $j \leftarrow j+1$. For the updated j value compute the component running sum $\gamma_m(r,j)$ and let $\Gamma_m^{'} = \Gamma_m - \{\gamma_{\max}\}$. If $\gamma_m(r,j)=0$ then let $\Gamma_{m+1} = \Gamma_m^{'}$. The decomposition of q_r is now complete and contains j labels, and if $\Gamma_{m+1} = \phi$ then END. Otherwise, i.e., if $\gamma_m(r,j) > 0$ then let $\Gamma_{m+1} = \Gamma_m^{'} \cup \{\gamma_m(r,j)\}$.
5) Let $\gamma_{m+1}(i,j) = \gamma_{\max} \in \Gamma_{m+1}$. Let $i=r$.
6) Go to step 3.

Example 1: Let us simulate the probability distribution $Q=(5/9, 4/9)$ from the biased coin with probability distribution $\Pi_2 = (2/3, 1/3)$. Applying the MAX-ENT algorithm described earlier we obtain

$$5/9 = 4/9 + \sum_{i=0}^{\infty} 8/3^{4+2i} \quad (1)$$

$$4/9 = 1/3 + 2/27 + \sum_{i=0}^{\infty} 8/3^{5+2i}. \quad (2)$$

It follows that $5/9$ is represented by the sequences $\{00, 0100, 010110, \ldots, 0101111\ldots110, \ \ldots\}$, and that $4/9$ is represented by the sequences $\{1, 011, 01010, 0101110, \ldots, 0101111 \ \ldots 1110, \ldots\}$. The leaf entropy is $H(V) = 0.991$ and the average sequence length is $E(W) = 2.185$. We notice that the binary coding expansion rate is $E(W) - H(Q) = 1.194$ bits.

Example 2: Let Q be the $K=2$ discrete probability distribution $(80/81, 1/81)$. We consider simulating Q from the biased coin probability distribution $(2/3, 1/3)$. Applying the MAX-ENT algorithm described earlier we obtain the sequence of running sum sets shown in Table I.

TABLE I
RUNNING SUM SETS AND LABELS FOR EXAMPLE 2

Running sum sets	Labels
$\Gamma_1 = \{\gamma_1(1), \gamma_1(2)\} = \{80/81, 1/81\}$	
$\Gamma_2 = \{\gamma_2(1), \gamma_2(2)\} = \{26/81, 1/81\}$	$v(1,1)=0$
$\Gamma_3 = \{\gamma_3(1), \gamma_3(2)\} = \{8/81, 1/81\}$	$v(1,2)=10$
$\Gamma_4 = \{\gamma_4(1), \gamma_4(2)\} = \{2/81, 1/81\}$	$v(1,3)=110$
$\Gamma_5 = \{\gamma_5(2)\} = \{1/81\}$	$v(1,4)=1110$
$\Gamma_6 = \phi$	$v(2,1)=1111.$

III. THE MINIMUM ENTROPY PER STEP ALGORITHM

We now describe the *minimum entropy per step* (MIN-ENT) algorithm. Many of the terms used here have already been introduced in Section II. At the m^{th} iteration, $m>1$, a label is assigned to a terminal node of the corresponding tree T_p, in a manner that the unused terminal node with largest probability P_m is assigned as a label to the probability q_r with minimum nonnegative value for the difference between its component running sum $\gamma_m(r,j)$ and P_m, i.e., such that $\min_i\{\gamma_m(i,j) - P_m | (\gamma_m(i,j) - P_m \geq 0)\} = \gamma_m(r,j) - P_m \geq 0,\ 1 \leq i \leq K$. The MIN-ENT algorithm consists of the following steps.

1) Let $m = 1$. Let $\gamma_1(i,1) = q_i,\ 1 \le i \le K$. Let $\Gamma_1 = \{q_1, q_2, \ldots, q_K\}$.
2) Determine $\gamma_{\max}$ and produce the tree T_p for the m^{th} iteration by expanding each terminal node in the tree from the m^{th} iteration, $m > 1$, whose probability exceeds $\gamma_{\max}$, by the least number of branches sufficient to make the resulting extended terminal node probability less than or equal to $\gamma_{\max}$.
3) Find the unused path E_l of length l in T_p whose probability is largest among not yet labeled paths, and denote this largest probability by P_m.
4) If $\min_i\{\gamma_m(i,j) - P_m | (\gamma_m(i,j) - P_m \ge 0)\} = \gamma_m(r,j) - P_m \ge 0$, then we associate to q_r the label (terminal node) $v(r,j)$ and the binary sequence of length l, whose digits constitute the labeling of E_l in T_p. This implies $\alpha(r,j) = P_m$. Compute the component running sum $\gamma_m^{'}(r,j)$ after this decomposition and let $\Gamma_m^{'} = \Gamma_m - \{\gamma_m(r,j)\}$. If $\gamma_m^{'}(r,j) = 0$ then let $\Gamma_{m+1} = \Gamma_m^{'}$. The decomposition of q_r has been obtained and contains j labels, and if $\Gamma_{m+1} = \phi$ then END. Otherwise, i.e., if $\gamma_m^{'}(r,j) > 0$, then let $\Gamma_{m+1} = \Gamma_m^{'} \cup \{\gamma_m^{'}(r,j)\}$.
5) Let $m \leftarrow m + 1$.
6) Go to step 2.

Example 3: Let $Q = \{53/81, 16/81, 4/27\}$. We consider generating Q, an operation equivalent to the perfect binary-constrained homophonic coding of a 3-ary source with symbol probability distribution Q, when $\Pi_2 = \{2/3, 1/3\}$ is the biased coin probability distribution. Applying the MAX-ENT algorithm we obtain

$$\begin{aligned} q_1 = \ 53/81 \ &= 4/9 + 4/27 + 4/81 + 2/243 \\ &\quad + \sum_{i=0}^{\infty} 8/3^{7+2i} \\ q_2 = \ 16/81 \ &= 4/27 + 4/81 \\ q_3 = \ 4/27 \ &= 1/9 + 2/81 + \sum_{i=0}^{\infty} 8/3^{6+2i}, \end{aligned}$$

which lead to an average sequence length $E(W) = 214/81$ and to a binary coding expansion rate $E(W) - H(Q) = 2.642 - 1.27 = 1.372$ bits. On the other hand, by using the MIN-ENT algorithm we obtain

$$\begin{aligned} q_1 = \ 53/81 \ &= 4/9 + 1/9 + 2/27 + 2/81 \\ q_2 = \ 16/81 \ &= 4/27 + 4/81 \\ q_3 = \ 4/27, & \end{aligned}$$

which lead to an average sequence length $E(W) = 68/27$ and to a binary coding expansion rate $E(W) - H(Q) = 2.52 - 1.27 = 1.25$ bits, i.e., a coding expansion rate representing 91% of the coding expansion rate obtained with the MAX-ENT algorithm.

Example 4: Let Q be the $K = 2$ discrete probability distribution with $q_1 = 1-(1-p)^n = 1-\overline{p}^n$ and $q_2 = (1-p)^n = \overline{p}^n$. We consider the generation of Q when $\Pi_2 = \{p, 1-p\}$ is the biased coin probability distribution. Applying the MIN-ENT algorithm we obtain the following probabilities for the labels representing q_1: $\alpha(1,1) = p,\ \alpha(1,2) = \overline{p}p,\ \alpha(1,3) = \overline{p}^2 p,\ \ldots,\ \alpha(1,j) = \overline{p}^{(j-1)}p, \ldots, \alpha(1,n) = \overline{p}^{(n-1)}p$, and for q_2 we obtain a single label $v(2,1)$ whose probability is $\alpha(2,1) = \overline{p}^n$. It follows that $H(V|Q = q_1) = -H(Q)/(1 - \overline{p}^n) + h(p)/p$, $H(V|Q = q_2) = 0$ and thus

$$\begin{aligned} H(V|Q) = & \\ & q_1 H(V|Q = q_1) + q_2 H(V|Q = q_2) \\ & = (1 - \overline{p}^n) H(V|Q = q_1) \\ & = -H(Q) + (1 - \overline{p}^n)\frac{h(p)}{p}, \end{aligned} \tag{3}$$

where $h(p) = -p\log p - (1-p)\log(1-p)$ is the binary entropy function [5]. However, since

$$H(V) = H(Q) + H(V|Q) = (1 - \overline{p}^n)h(p)/p,$$

it follows from (3) that

$$\lim_{n\to\infty} H(V|Q) = \frac{h(p)}{p},$$

because $\lim_{n\to\infty} H(Q) = 0$. We remark that both the MAX-ENT algorithm and the MIN-ENT algorithm produce identical results in this example because at each step of either algorithm there is only one possibility for performing the probability expansion, i.e., $\gamma_m(1) - P_m > 0$ and $\gamma_m(2) - P_m < 0$, for $1 \le m \le n$. For $m = n + 1$ we have $\gamma_{n+1}(1) = 0$ and $\gamma_{n+1}(2) - P_{n+1} = 0$.

IV. Using Two or More Biased Coins

In this section we introduce a generalization of the MIN-ENT algorithm [17], by generating a discrete probability distribution using two or more biased coins, obtaining results similar to Gargano and Vaccaro's [15], with the distinction of not necessarily using a probability distribution of heads and tails dependent on n. We introduce next some notation that will be used in the sequel.

A. Basic Terminology

A tree T is used to indicate the choice of distinct coins by the algorithm in order to produce the desired probability distribution. A distinct labeled leaf in T is associated one to one with each one of the possible outcomes. Given an algorithm to generate an n-sided die and its associated tree T, the worst case bounded time $L_{\max}$ to produce an outcome is given by

$$L_{\max} = \max_x l_T(x), \tag{4}$$

and the average time is given by

$$E[T] = \sum_x p(x) l_T(x), \tag{5}$$

where $l_T(x)$ denotes the depth level of x in T, i.e., the length of the path from the root of T to the leaf x, and $p(x)$ denotes the probability of a leaf x being reached.

In what follows we assume that, for each tree considered, two branches from each node emanate. Each node is labeled using a coin distribution. If a node has no indication we assume the coin used is an unbiased coin, and if the node is indicated by $(p, 1-p)$ this means a biased coin is used

with head's probability given by p and tail's probability given by $1-p$. Each branch is labeled with a probability. Following [15], for any positive integer n let $p(n)$ be defined as

$$p(n) = \max\{i : 2^i \text{ divides } n\}, \tag{6}$$

and let $r(n) = n - 2^{\lfloor \log n \rfloor}/2^{p(n)}$. It follows from (6) that $p(n) = 0$ if n is odd. The biased coin, as suggested in [15], has probability distribution (P_H, P_T) for heads and tails given by

$$(P_H, P_T) = \left(2^{\lceil \log r(m) \rceil}/m, 1 - 2^{\lceil \log r(m) \rceil}/m\right), \tag{7}$$

where m is the largest odd factor of n. In order to generate a fair die with n faces, in the worst case [15, Theorem 1]

$$1 + \lfloor \log n \rfloor + \lceil \log(n - 2^{\lfloor \log n \rfloor}) \rceil + p(n) \tag{8}$$

coin flips are required, and on average

$$1 + \lfloor \log n \rfloor + (2^{p(n)}/n)(\lceil \log r(n) \rceil 2^{\lceil \log r(n) \rceil} - r(n)(\lfloor \log r(n) \rfloor - p(n))), \tag{9}$$

coin flips are required.

B. Description of the Algorithm

The algorithm proposed here follows essentially the same steps of the one introduced in [17] with the important difference that instead of using only one coin we use two or more coins and at each step we decide which coin must be chosen to be flipped, in order to minimize the entropy in that step.

Let $m_1 = \{p_1, 1-p_1\}$, $m_2 = \{p_2, 1-p_2\}$, ..., $m_r = \{p_r, 1-p_r\}$ denote the probability distribution of coins $1, 2, \ldots, r$, respectively, and let $Q = \{q_1, q_2, \ldots, q_K\}$ denote the probability distribution to be generated, i.e, the target probability distribution, where K is a positive integer, $K \geq 2$. For a given source, our algorithm, henceforth referred to as the Λ-algorithm, finds the decomposition $p_1^{\lambda_1} p_2^{\lambda_2} \ldots p_r^{\lambda_r} (1-p_1)^{l_1-\lambda_1}(1-p_2)^{l_2-\lambda_2} \ldots (1-p_r)^{l_r-\lambda_r}$ of each probability in Q as a sum (finite or infinite) of terms, where λ_i denotes the number of heads and $l_i - \lambda_i$ denotes the number of tails, $0 \leq i \leq r$, for the coin with probability distribution m_i, and $\sum_{i=1}^{r} l_i = l$ for a sequence of length l.

Definition 5: For a given finite set of biased coins with probability distribution $m_1 = \{p_1, 1-p_1\}$, $m_2 = \{p_2, 1-p_2\}$, ..., $m_r = \{p_r, 1-p_r\}$, we define a hybrid binary tree as a binary tree for which each node is associated with one of the probability distributions $m_i,\ 1 \leq i \leq r$.

The Λ-algorithm, when applied to the target probability distribution, generates a hybrid tree T where each leaf in T is associated with a probability in the target probability distribution. The probability of a path of length l in T, containing $\lambda_1+\lambda_2+\ldots+\lambda_r$ heads and $(l_1-\lambda_1)+(l_2-\lambda_2)+\ldots+(l_r-\lambda_r)$ tails, is $p_1^{\lambda_1} p_2^{\lambda_2} \ldots p_r^{\lambda_r}\ (1-p_1)^{l_1-\lambda_1}(1-p_2)^{l_2-\lambda_2} \ldots (1-p_r)^{l_r-\lambda_r}$. In particular, for computing the probability of a leaf (terminal node), the path extending from the root node to that terminal node is considered. For $m = 1$ grow trees $T_1, T_2, \ldots, T_r$ from the root, generated from the flip of the coins associated with the probability distributions $m_1, m_2, \ldots, m_r$, respectively. Expand by one branch each terminal node in those trees whose probability exceeds $\gamma_{\max}$. Keep only those trees for which at least one resulting extended terminal node probability is less than or equal to $\gamma_{\max}$. The resulting s trees are called *processed binary rooted trees with probabilities*, $T_{p_\ell}, 1 \leq \ell \leq s, s \leq r$.

At the m^{th} iteration, $m > 1$, the minimum nonnegative value is computed for the difference between the running sums in the running sum set Γ_m and P_m, i.e., $\min_i\{\gamma_m(i,j) - P_m | (\gamma_m(i,j) - P_m \geq 0)\} = \gamma_m(t,j) - P_m \geq 0$, where P_m denotes the largest probability of a not yet labeled terminal node, among all $T_{p_\ell},\ 1 \leq \ell \leq s$. Such a terminal node is assigned to q_t. Notice that at each iteration for $m > 1$ there is a Γ_{m,p_ℓ} associated to each one of the T_{p_ℓ} trees. The norm of Γ_{m,p_ℓ} is given by

$$\|\Gamma_{m,p_\ell}\| = \sqrt{\sum_{i=1}^{K} (\gamma_{m,p_\ell}(i,j))^2}. \tag{10}$$

and its minimum value is used to establish a criterium to choose the trees that will be kept for the following iteration.

Since the size of surviving trees grow exponentially with m, a rule is desired to eliminate those surviving trees for which their respective running sums at the m^{th} step do not satisfy some convergence criteria. Therefore, a value for $L_{\max}$ is chosen, being denoted by M.

The Λ-algorithm consists of the following steps.

1) Let $m = 1$. Let $\gamma_1(i,1) = q_i,\ 1 \leq i \leq K$. Let $\Gamma_1 = \{q_1, q_2, \ldots, q_K\}$. Grow each tree $T_1, T_2, \ldots, T_r$ from the root node by a depth of one according to their respective coin probability distributions $m_1, m_2, \ldots, m_r$, respectively.
2) Determine $\gamma_{\max}$ and produce the trees $T_{p_\ell}, 1 \leq \ell \leq s$, for the m^{th} iteration by expanding, by a depth of one, each terminal node in each T_{p_ℓ} tree from the $(m-1)^{th}$ iteration, $m > 1$, whose probability exceeds $\gamma_{\max}$, and by keeping those expanded trees for which at least one extended terminal node probability (leaf probability) is less than or equal to $\gamma_{\max}$.
3) Calculate the norm for each tree T_{p_ℓ} using (10). Keep the tree(s) with smallest $\|\Gamma_{m,p_\ell}\|$.
4) For each $\ell,\ 1 \leq \ell \leq s$, find the not yet labeled path E_{l_ℓ} of length l in T_{p_ℓ} whose probability P_{m_ℓ} is largest among unused paths. Denote by P_m the largest probability among all P_{m_ℓ} that do not exceed $\gamma_{\max}$, and denote by E_l the respective path.
5) If $\min_i\{\gamma_m(i,j) - P_m | (\gamma_m(i,j) - P_m \geq 0)\} = \gamma_m(t,j) - P_m \geq 0,\ 1 \leq i \leq K$, then associate to q_t the terminal node $v(t,j)$. This implies $\alpha(t,j) = P_m$. Compute the running sum $\gamma_m^{'}(t,j)$ after this decomposition and let $\Gamma_m^{'} = \Gamma_m - \{\gamma_m(t,j)\}$. If $\gamma_m^{'}(t,j) = 0$ then let $\Gamma_{m+1} = \Gamma_m^{'}$. The decomposition of q_t is now complete and contains j terms, and if $\Gamma_{m+1} = \phi$ then END. Otherwise, i.e., if $\gamma_m^{'}(t,j) > 0$, then let $\Gamma_{m+1} = \Gamma_m^{'} \cup \{\gamma_m^{'}(t,j)\}$.
6) Let $m \leftarrow m + 1$.
7) If $m = M$ stop, otherwise go to step 2.

C. Performance Comparison

In this *Section* we provide examples showing smaller values for the average time $E[T]$ than those in [15]. Using the Λ-

algorithm, there are cases where the maximal length $L_{\max}$ is not bounded but even in these cases a shorter $E[T]$ results. The Λ-algorithm is a generalization for more than one biased coin of the algorithm introduced in [17], obtaining equivalent and in some cases even better results than those in [15]. In order to compare the performance of the Λ-algorithm with that of Gargano and Vaccaro [15], we are going to use the same coins that would be used in [15] for the generation of a uniform probability distribution using two coins, one fair and the other with distribution given by (7). We call attention once more to the fact that the biased coins employed in the Λ-algorithm are arbitrary, i.e., they do not depend on n as is the case for the algorithm in [15]. The biased coin, as suggested in [15], has probability distribution for heads P_H, and tails P_T, given by (7).

Example 5: Consider generating the probability distribution of the faces of a fair die, i.e., $n = 6$, using the coins with probability distribution $m_1 = (1/2, 1/2)$ and $m_2 = (2/3, 1/3)$, respectively.

Since two coins, i.e., $r = 2$, are employed it follows from the Λ-algorithm, step 1, that two trees are generated, namely tree $T1$ associated with coin m_1, and tree $T2$ associated with coin m_2, as illustrated in Figure 1. We notice that all branch probabilities in both $T1$ and $T2$ (Figure 1) are greater than $1/6$, so by the Λ-algorithm, step 3, it is necessary to grow each tree taking into account the use of coins m_1 and m_2. The resulting expanded hybrid trees are illustrated in Figure 2 and in Figure 3, respectively. Notice that tree (a) in Figure 2 has all branch probabilities greater than $1/6$, while the other trees in that figure as well as those in Figure 3 present at least one branch probability less than or equal to $1/6$. Therefore, by the Λ-algorithm, step 3, tree (a) in Figure 2 is eliminated while trees (b), (c) and (d) are kept for the next step. For the trees (b), (c) and (d) in Figure 2 and the trees (a), (b), (c) and (d) in Figure 3, we have the following running sum sets

$$\begin{aligned}
\Gamma_{1,1} &= \{0, 0, 1/6, 1/6, 1/6, 1/6\} \\
\Gamma_{1,2} &= \{0, 1/6, 1/6, 1/6, 1/6, 1/6\} \\
\Gamma_{1,3} &= \{0, 1/6, 1/6, 1/6, 1/6, 1/6\} \\
\Gamma_{1,4} &= \{0, 0, 1/6, 1/6, 1/6, 1/6\} \\
\Gamma_{1,5} &= \{1/18, 1/6, 1/6, 1/6, 1/6, 1/6\} \\
\Gamma_{1,6} &= \{1/18, 1/6, 1/6, 1/6, 1/6, 1/6\} \\
\Gamma_{1,7} &= \{0, 0, 1/6, 1/6, 1/6, 1/6\},
\end{aligned}$$

and the norms associated to each one are as follows.

$$\begin{aligned}
\|\Gamma_{1,1}\| &= \sqrt{4(1/6)^2} = 0.33 \\
\|\Gamma_{1,2}\| &= \sqrt{5(1/6)^2} = 0.38 \\
\|\Gamma_{1,3}\| &= \sqrt{5(1/6)^2} = 0.38 \\
\|\Gamma_{1,4}\| &= \sqrt{4(1/6)^2} = 0.33 \\
\|\Gamma_{1,5}\| &= \sqrt{(1/18)^2 + 5(1/6)^2} = 0.38 \\
\|\Gamma_{1,6}\| &= \sqrt{(1/18)^2 + 5(1/6)^2} = 0.38 \\
\|\Gamma_{1,7}\| &= \sqrt{4(1/6)^2} = 0.33.
\end{aligned}$$

By step 3 of the Λ-algorithm, tree (b) illustrated in Figure 2, and trees (a) and (d) illustrated in Figure 3 are kept.

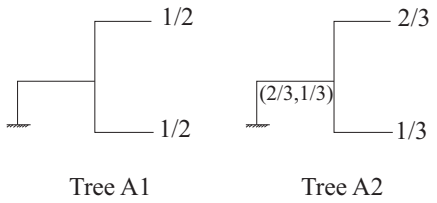

Fig. 1. Trees T1 and T2 generated by the first flip of coins with probability distribution m_1 and m_2, respectively.

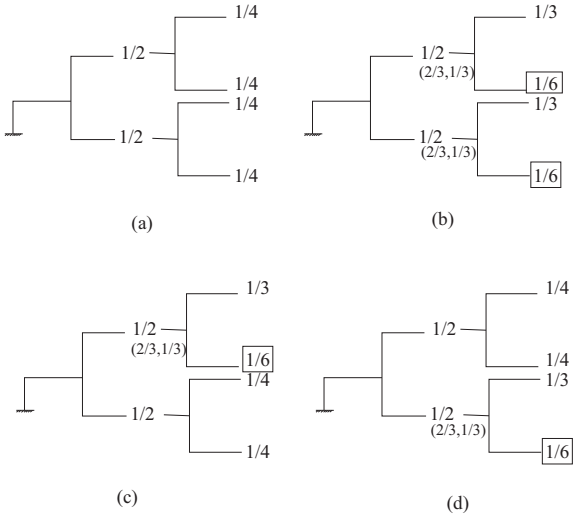

Fig. 2. All possibilities of growth for tree A1 in two steps of the Λ-algorithm.

In the next step only the surviving trees are considered and the branches with probability greater than $1/6$ must be expanded, remembering always to consider all the possible expansions using coins m_1 and m_2. Two of the trees that are obtained by applying the Λ-algorithm to depth three from the root node are shown in Figures 4 and 5.

The trees shown in Figures 4 and 5 provide distinct solutions to the problem of generating a uniform probability distribution for $n = 6$, and both are bounded trees. We notice that the tree in Figure 4 is the same as that which results by using the

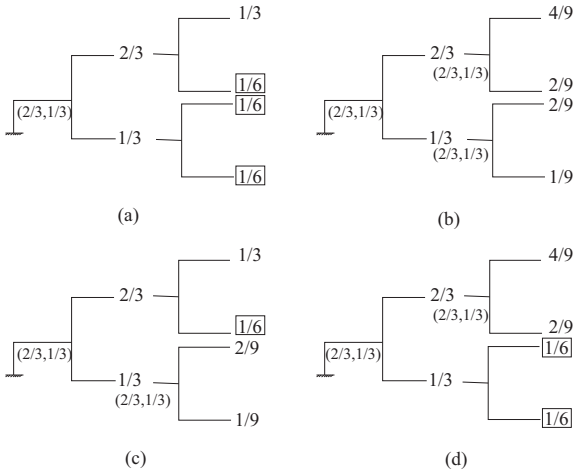

Fig. 3. All possibilities of growth for tree A2 in two steps of the Λ-algorithm.

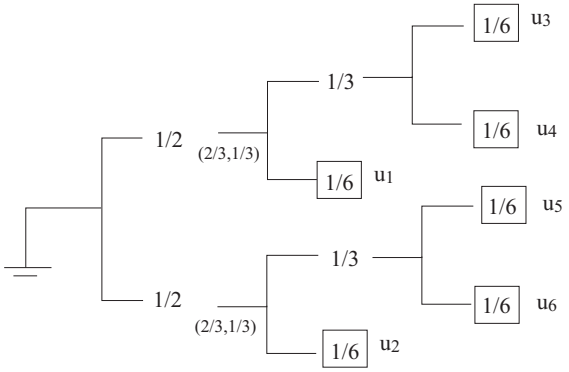

Fig. 4. Tree $T1$ obtained for $n = 6$ coincides with that for the algorithm in [15].

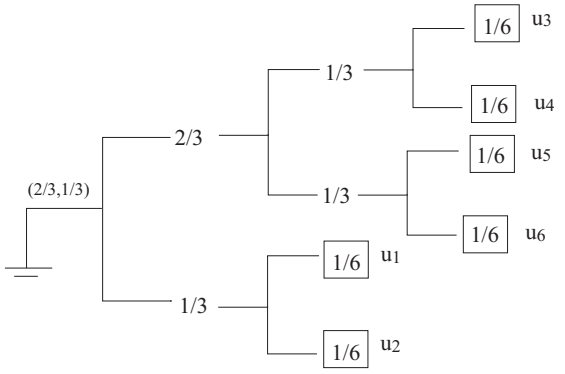

Fig. 5. Tree A2 obtained for $n = 6$.

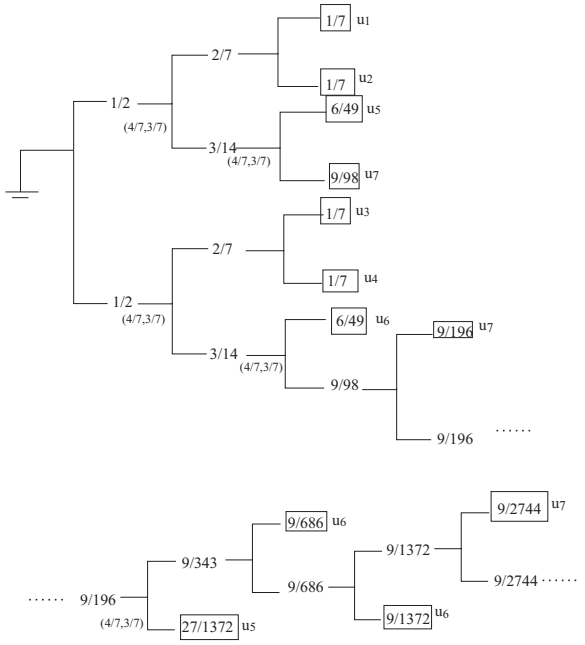

Fig. 6. Tree obtained for $n = 7$ using the Λ-algorithm.

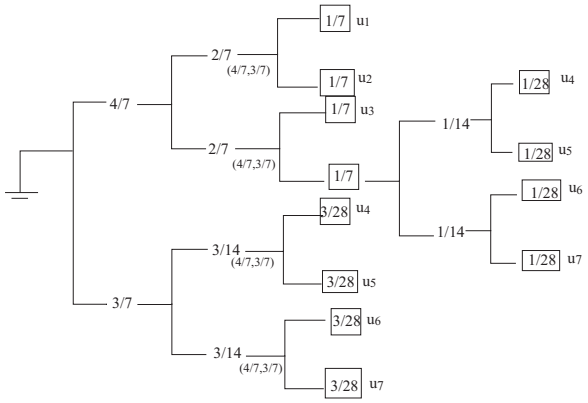

Fig. 7. Tree obtained for $n = 7$ using the algorithm in [15].

algorithm in [15]. Applying (4) and (5) to the tree in Figure 4 the values $L_{\max} = 3$ and $E[T] = 2.67$ result for the worst case time and for average time, respectively. Identical results for $L_{\max}$ and $E[T]$ are obtained for the tree in Figure 5.

Example 6: Consider generating a uniform probability distribution for a random variable with $n = 7$ possible outcomes, using the coins with probability distribution $m_1 = (1/2, 1/2)$ and $m_2 = (3/7, 4/7)$, respectively. One of the trees obtained using the Λ-algorithm for $n = 7$ is illustrated in Figure 6, and the tree obtained by the use of the algorithm introduced in [15] is shown in Figure 7.

It should be noticed that the parameter $L_{\max}$ for the tree in Figure 7 is bounded. On the other hand using the Λ-algorithm, $L_{\max}$ in this example is unbounded (Figure 6) but the resulting average time is $E[T] = 3.1902$, and is a better result than $E[T] = 3.29$, obtained when the algorithm in [15] is used.

V. Conclusions

A new algorithm for the generation of a string of random variables drawn from a discrete probability distribution using the flips of two or more coins, some of them biased, was introduced. In particular, this approach contributes an alternative solution to the classical problem of generating a discrete uniform probability distribution using two or more unbiased coins. It was shown by some simple examples with the Λ-algorithm that it is possible to obtain equivalent and in some cases even better results than those in [15]. In principle, the choice of coins that must be employed to generate a discrete probability distribution using the Λ-algorithm does not depend on n. However, some examples have shown that the performance of the Λ-algorithm varies with the biased coins that are chosen, i.e., the expected length of the labels can vary. For this reason, it is important to investigate a criterium for specifying coin probability distributions, aiming at the optimization of the Λ-algorithm. Another point for future research is the investigation of ways to limit $L_{\max}$ in those cases where the tree produced by the Λ-algorithm is unbounded.

Acknowledgment

The authors are grateful to the Editor and to the reviewers for providing constructive comments that have improved the quality of this paper. Danielle P. B. de Arruda Camara acknowledges partial support from the Pernambuco State Foundation to Support Science and Technology - FACEPE, Project APQ-0055-3.04/09. Valdemar C. da Rocha Jr. and Cecilio Pimentel acknowledge partial support from the Brazilian

National Council for Scientific and Technological Development - CNPq, Project No. 304696/2010-2 and Project No. 303884/2013-4, respectively.

REFERENCES

[1] D. E. Knuth, *The Art of Computer Programming*, Volume 2, Seminumerical Algorithms, Addison-Wesley, Reading Mass., 688 pages, 1981. doi: http://dx.doi.org/10.1002/spe.4380120909.

[2] J. von Neumann, "Various techniques used in connection with random digits, notes by G. E. Forsythe, National Bureau of Standards," Applied Math Ser., vol. 12, pp. 36-38; reprinted in von Neumann's Collected Works., vol. 5. Oxford, U.K.: Pergamon, pp. 768-770, 1963.

[3] J. Abrahams, "Generation of discrete distributions from biased coins," *IEEE Trans. on Inform. Theory*, vol. 42, no. 5, pp. 1541-1546, Sept. 1996. doi: http://dx.doi.org/10.1109/18.532895.

[4] M. Blum, "Independent unbiased coin flips from a correlated biased source-A finite state Markov chain," *Combinatorica*, vol. 6, no. 2, pp. 97-108, 1986. doi: http://dx.doi.org/10.1007/BF02579167.

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[6] E. W. Dijkstra, "Making a fair roulette from a possibly biased coin," *Inform. Processing Lett.*, vol. 36, p. 193, 1990. doi: http://dx.doi.org/10.1016/0020-0190(90)90072-6.

[7] P. Elias, "The efficient computation of an unbiased random sequence," *Ann. Math. Statist.*, vol. 43, pp. 865-870, 1972.

[8] W. Hoeffding and G. Simmons, "Unbiased coin tossing with a biased coin," *Ann. Math. Statist.*, vol. 41, pp. 341-352, 1970.

[9] T. S. Han and M. Hoshi, "Interval algorithm for random number generation," *IEEE Trans. on Inform. Theory*, vol. 43, no. 2, pp. 599-611, March 1997. doi: http://dx.doi.org/10.1109/18.556116.

[10] Y. Horibe, "Entropy and optimal random number transformation," *IEEE Trans. on Inform. Theory*, vol. 27, no. 4, pp. 527-529, July 1981. doi: http://dx.doi.org/10.1109/TIT.1981.1056363.

[11] D. E. Knuth and A. C. Yao, "The complexity of nonuniform random number generation," in Algorithms and Complexity, New Directions and Results, J. F. Traub, Ed. New York: Academic, pp. 357-428, 1976.

[12] R. Motwani and P. Raghavan, *Randomized Algorithms. Cambridge*, U.K.: Cambridge Univ. Press, 1996.

[13] Q. F. Stout and B. Warren, "Tree algorithms for unbiased coin tossing with a biased coin," *Ann. Probab.*, vol. 12, pp. 212-222, 1984. doi: http://dx.doi.org/10.1214/aop/1176993384.

[14] S. Pae, *Random Number Generation Using a Biased Source*, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 2005.

[15] L. Gargano and U. Vaccaro, "Efficient generation of fair dice with few biased coins," *IEEE Trans. on Inform. Theory*, vol. 45, no. 5, pp. 1600-1606, July 1999. doi: http://dx.doi.org/10.1109/18.771228.

[16] V. C. da Rocha Jr. and C. Pimentel, "Binary-constrained homophonic coding," *VI International Symposium on Communication Theory & Applications*, 15 - 20 July 2001, Ambleside, England, pp. 263-268.

[17] V. C. da Rocha Jr. and C. Pimentel, "Optimum binary-constrained homophonic coding," *VII International Symposium on Communication Theory and Applications*, 13 - 18 July 2003, Ambleside, England, pp. 64-69.

[18] D. Feldman, R. Impagliazzo, M. Naor, N. Nisan, N., S. Rudich and A. Shamir, "On dice and coins: Models of computation for random generation," *Inform. Comput.*, vol. 104, pp. 159-174, 1993. doi: http://dx.doi.org/10.1006/inco.1993.1028.

[19] H. Zhou and J. Bruck, "Efficient generation of random bits from finite state Markov chains," *IEEE Trans. Inform. Theory*, vol. 58, pp. 2490 - 2506, Apr. 2012. doi: http://dx.doi.org/10.1109/TIT.2011.2175698.

[20] H. Zhou and J. Bruck, "A universal scheme for transforming binary algorithms to generate random bits from loaded dice," ArXiv:1209.0726 [cs.IT], Sept. 2012.

[21] H. Zhou and J. Bruck, "Variable-length extractors," in *Proc. IEEE International Symposium on Information Theory* (ISIT), 2012, Cambridge, USA, pp.1107-1111. doi: http://dx.doi.org/10.1109/ISIT.2012.6283024.

[22] H. Zhou and J. Bruck, "Streaming algorithms for optimal generation of random bits," ArXiv:1209.0730 [cs.IT], Sep. 2012.

[23] S. Pae, "Exact output rate of generalized Peres algorithm for generating random bits from loaded dice," *International Journal of Advanced Computer Science and Applications* (IJACSA), Vol. 4, No.7, pp. 218-221, 2013. doi:10.14569/IJACSA.2013.040731.

[24] S. Pae, "Exact output rate of Peres's algorithm for random number generation," *Inf. Process. Lett.*, vol. 113, no. 5-6, pp. 160 - 164, March 2013. doi: http://dx.doi.org/10.1016/j.ipl.2012.12.012.

Valdemar C. da Rocha Jr. (M'77, SM'04, LSM'13) was born in Jaboatão, Pernambuco, Brazil, on August 27, 1947. He received in 1970 the B.Sc. degree in Electrical/Electronics Engineering from the Escola Politécnica, Recife, Brazil, and in 1976 he received the Ph.D. degree in Electronics from the University of Kent at Canterbury, U.K. He joined the faculty of the Federal University of Pernambuco, Recife, Brazil, in 1976 as an Associate Professor and founded its Electrical Engineering Postgraduate Programme. He served as Department Chair (1992-1996), and in 1993 he became Professor of Telecommunications.

He was editor for Coding Theory and Techniques, Journal of Communication and Information Systems, co-sponsored by the Brazilian Telecommunications Society and the IEEE Communications Society, and has been a reviewer for a number of scientific journals including IET Electronics Letters, IET Communications and IEEE Transactions on Information Theory. He has also been involved in the organization of conferences in Brazil and abroad.

He is a founder (2002) and past President (2002-2004) of the IEEE Information Theory Society Chapter, Brazil Council. He is founder (2003) and Vice-President for three consecutive terms (2003-2015) of the Institute for Advanced Studies in Communications. He is a founding member (1983) of the Brazilian Telecommunications Society, served as Vice-President for two terms (2000-2004) and as President also for two terms (2004-2008). He joined the IEEE Communications Society in 1977 and the IEEE Information Theory Society in 1981. He is a Member (1982) of the Brazilian Society of Applied and Computational Mathematics, and a Fellow (1992) of the Institute of Mathematics and its Applications, UK.

During 1990-1992, he was a Visiting Professor at the Swiss Federal Institute of Technology-Zurich, Institute for Signal and Information Processing. In 2005-2006 he was a Visiting Professor at the Institute of Integrated Information Systems, University of Leeds, UK, and in 2007 he was a Visiting Professor at the Department of Communication Systems, Lancaster University, UK.

Prof. da Rocha research interests are in applied digital information theory, including error-correcting codes and cryptography. He has published over 100 engineering and scientific papers, including journal and conference papers, and the books Communication Systems, Springer, 2005, and Elements of Algebraic Coding Systems, Momentum Press, 2014.

Danielle Paes Barretto de Arruda Camara was born in Recife, Pernambuco, Brazil, on June 9, 1974. She received the B.Sc. (1998), M.Sc. (2001) and Ph.D. (2006) degrees, all in Electrical Engineering from the Federal University of Pernambuco, Recife, Brazil. She conducted her post-doctoral research in France (2007-2008) where she participated in the Biotyful project concerned with the combination of biometrics and cryptography with the Intermedia group of TELECOM SudParis. This multidisciplinary project that involved research areas such as cryptography, biometrics, error-correcting codes and information theory was funded by the French National Agency for Research - ANR.

She developed a three year project (2009-2012) entitled "Cryptographic Security based on Noisy Physical Elements" for regional scientific development in Brazil which was supported by the Pernambuco State Foundation to Support Science and Technology - FACEPE and by the Brazilian National Council for Scientific and Technological Development - CNPq.

She joined the IEEE Information Theory Society in 2007, the IEEE Computer Society and IEEE Biometrics Council in 2013. She is an IEEE Certified Biometrics Professional (CBP) since 2010. Dr. Camara's professional experience includes research, project coordination, teaching, undergraduate project advising. During her master and doctorate she was a teaching assistant for courses on number theory, cryptography and information theory. Also during her doctorate she worked as a part-time lecturer teaching courses on stochastic processes, scientific methodology and cryptography at University of Pernambuco, Recife, Brazil. She also worked as part-time lecturer teaching courses on linear algebra and analytical geometry, mathematical logic and optical telecommunications systems at Integrated Faculties of Recife (FIR), Recife, Brazil.

Dr. Camara's research interests includes cryptography, information theory, error-correcting codes and biometrics.

Cecilio Pimentel (M'97, SM'13) was born in Recife, Brazil, in 1966. He received the B.Sc. degree from the Federal University of Pernambuco, Recife, Brazil, in 1987; the M.Sc. degree from the Catholics University of Rio de Janeiro, Rio de Janeiro, Brazil, in 1990; and the Ph.D. degree from the University of Waterloo, Ontario, Canada, in 1996, all in electrical engineering. Since October 1996, he has been with the Department of Electronics and Systems at the Federal University of Pernambuco, where he is currently an Associate Professor. From 2007 to 2008, he was a Visiting Research Scholar at the Department of Mathematics and Statistics, Queen's University, Kingston, Canada. His research interests include digital communications, information theory, and error correcting coding. He is a Senior Member of the Brazilian Telecommunications Society (SBrT) and is a Senior Member of the IEEE.